UNIVERSITY OF BIRMINGHAM
University of Birmingham Research at Birmingham

# Learning modular and transferable forward models of the motions of push manipulated objects

Kopicki, Marek; Zurek, Sebastian; Stolkin, Rustam; Moerwald, Thomas; Wyatt, Jeremy L.

*DOI:*
10.1007/s10514-016-9571-3

*License:*
Creative Commons: Attribution (CC BY)

*Document Version*
Publisher's PDF, also known as Version of record

*Citation for published version (Harvard):*
Kopicki, M, Zurek, S, Stolkin, R, Moerwald, T & Wyatt, JL 2017, 'Learning modular and transferable forward models of the motions of push manipulated objects', *Autonomous Robots*, vol. 41, no. 5, pp. 1061–1082. https://doi.org/10.1007/s10514-016-9571-3

Link to publication on Research at Birmingham portal

**General rights**
Unless a licence is specified above, all rights (including copyright and moral rights) in this document are retained by the authors and/or the copyright holders. The express permission of the copyright holder must be obtained for any use of this material other than for purposes permitted by law.

•Users may freely distribute the URL that is used to identify this publication.
•Users may download and/or print one copy of the publication from the University of Birmingham research portal for the purpose of private study or non-commercial research.
•User may use extracts from the document in line with the concept of 'fair dealing' under the Copyright, Designs and Patents Act 1988 (?)
•Users may not further distribute the material nor use it for the purposes of commercial gain.

Where a licence is displayed above, please note the terms and conditions of the licence govern your use of this document.

When citing, please reference the published version.

**Take down policy**
While the University of Birmingham exercises care and attention in making items available there are rare occasions when an item has been uploaded in error or has been deemed to be commercially or otherwise sensitive.

If you believe that this is the case for this document, please contact UBIRA@lists.bham.ac.uk providing details and we will remove access to the work immediately and investigate.

Download date: 05. May. 2023

Auton Robot
DOI 10.1007/s10514-016-9571-3

# Learning modular and transferable forward models of the motions of push manipulated objects

**Marek Kopicki**[1] · **Sebastian Zurek**[1] · **Rustam Stolkin**[1] · **Thomas Moerwald**[2] · **Jeremy L. Wyatt**[1]

Received: 19 October 2015 / Accepted: 13 May 2016
© The Author(s) 2016. This article is published with open access at Springerlink.com

**Abstract** The ability to predict how objects behave during manipulation is an important problem. Models informed by mechanics are powerful, but are hard to tune. An alternative is to learn a model of the object's motion from data, to *learn to predict*. We study this for push manipulation. The paper starts by formulating a quasi-static prediction problem. We then pose the problem of learning to predict in two different frameworks: (i) regression and (ii) density estimation. Our architecture is modular: many simple, object specific, and context specific predictors are learned. We show empirically that such predictors outperform a rigid body dynamics engine tuned on the same data. We then extend the density estimation approach using a product of experts. This allows transfer of learned motion models to objects of novel shape, and to novel actions. With the right representation and learning method, these transferred models can match the prediction performance of a rigid body dynamics engine for novel objects or actions.

**Keywords** Transfer learning · Manipulation · Prediction · Robot Learning

✉ Jeremy L. Wyatt
jlw@cs.bham.ac.uk

Marek Kopicki
msk@cs.bham.ac.uk

Thomas Moerwald
moerwald@acin.tuwien.ac.at

[1] School of Computer Science, University of Birmingham, Birmingham B15 2TT, UK

[2] ACIN, TU Wien, Gusshausstrasse 27-29 / E376, 1040 Vienna, Austria

## 1 Introduction

Prediction is central to intelligent behaviour. An agent must predict its actions' effects, so as to be able to plan to achieve goals (Craik 1943). This paper concerns prediction of the motions of a manipulated rigid object. Within this scope, the paper makes the following contributions. First, it presents a *modular machine learning* solution to object motion prediction problems, and shows that, when tuned for specific objects (or contexts), this approach outperforms physics simulation. Second, it shows transfer of learned motion models to novel objects and actions, with the first results shown for real objects. This ubiquity is precisely what rigid body simulators are designed to achieve. In this paper, we show that, if the right representations are used, transfer learning can even approach the prediction performance of a rigid body simulator on novel objects and actions.

This paper thus extends our previous work (Kopicki 2010; Kopicki et al. 2011), where the core prediction algorithm was presented, and tested mostly in simulation. This paper tests three specific hypotheses, all evaluated with respect to real objects. Hypothesis 1 (H1) is that a *modular learning* approach can outperform physics engines for prediction of rigid body motion. Hypothesis 2 (H2) is that by factorising these modular predictors they can be transferred to make predictions about novel actions. Hypothesis 3 (H3) is that by factorising, learning can be transferred to make predictions about novel shapes. We suppose that learning transfer is only effective given a good representation.

The paper is structured as follows. Section 2 gives a motivation and background. Section 3 describes the problems to be solved, and the modular learning approach. Next, Sect. 4 introduces representations of object motion, enabling a formal problem statement in Sect. 5 and formulation in both regression and density estimation frameworks. Section 6

Published online: 22 June 2016

incorporates contact information, and Sect. 7 describes how we can factor the learner by the contacts to achieve transfer learning. Section 8 gives implementation details, Sect. 9 the experimental method, and Sect. 10 the corresponding results. Section 11 reviews related work. We finish with a discussion in Sect. 12.

## 2 The importance of prediction for manipulation

The human motor system uses predictive (or forward) models of the effects that motor actions have on sensory state (Flanagan et al. 2003, 2006; Mehta and Schaal 2002; Witney et al. 2000; Johansson and Cole 1992). The predictions are used for a variety of purposes, including feedforward control, coordination of motor systems, action planning, and monitoring of action plan execution. Neuroscientists have highlighted their importance for dexterous manipulation.

Predictive models are also useful in robot manipulation. One approach is to build a model informed by theories of mechanics (Mason 1982; Lynch 1992; Lynch and Mason 1996; Peshkin and Sanderson 1988; Cappelleri et al. 2006; Mason 2001; Flickinger et al. 2015), to make predictions of robot and object motion under contact. Various analytic models exist, some making the quasi-static assumption, and others modelling dynamics. To be useful for manipulation planning, their predictions must be made over a variety of timescales. To make metrically precise predictions, these models require explicit representation of intrinsic parameters, such as friction, mass, mass distribution, and coefficients of restitution. These are not trivial to estimate. Even then, model approximations can cause inaccurate predictions. Despite these challenges, analytic approaches have promise, and much work on push planning uses either purely kinematic models (Stilman and Kuffner 2008), quasi static models (Dogar and Srinivasa 2010; Lynch and Mason 1996) or rigid body dynamics engines Zito et al. (2012), Cosgun et al. (2011).

A second approach is to learn a forward model. Most such models are of qualitative effects (Montesano et al. 2008; Moldovan et al. 2012; Hermans et al. 2011; Fitzpatrick et al. 2003; Ridge et al. 2010; Kroemer and Peters 2014), although metrically precise models have been learned (Meriçli et al. 2014; Scholz and Stilman 2010). These learn action-effect correlations. We extend this to learning full rigid body motions in SE(3), in which objects may twist, slide, topple, and make and break contacts with both the robot and the environment. To achieve this we propose a modular approach.

## 3 Three prediction problems

To understand hypotheses H1–H3, consider three corresponding prediction problems in Fig. 1a–c: action interpolation (P1), action transfer (P2) and shape transfer (P3). Consider how each might be tackled using either: (i) rigid body simulator employing classical mechanics or (ii) statistical machine learning. Assume object and environment shape to be known.

### 3.1 Action interpolation (P1)

Suppose some pushes of an object were made (Fig. 1a top row). In some cases the object tipped, in some it slid. Now a new push direction is tried (bottom row, left column). The task is to predict the new object motion. To do this, rigid body mechanics requires parameters such as the object mass and frictional coefficients. These may be estimated

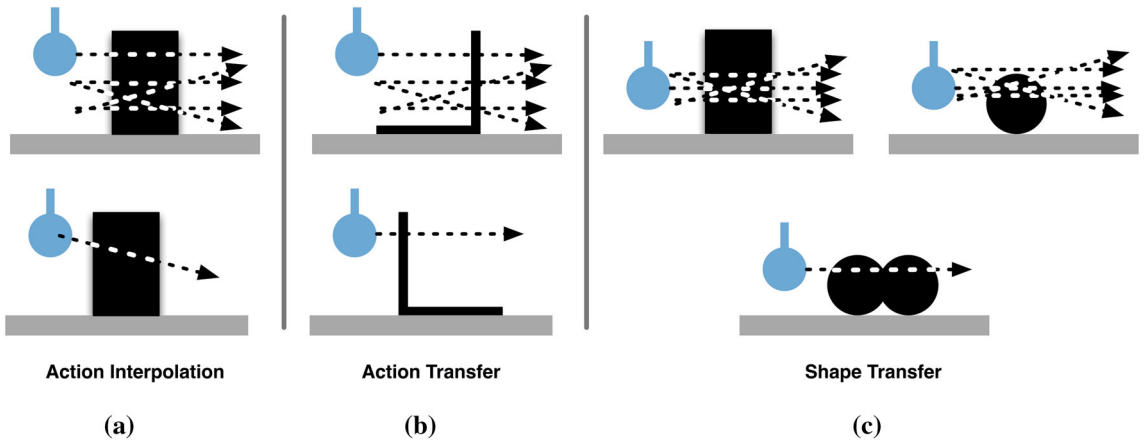



**Fig. 1** Three types of prediction problem. A robot finger is shown in *blue*, objects in *black*, and motions of the finger as dashed lines with *arrows*. *Top row* training actions. *Bottom row* an example test action. Each *column* represents a different problem. **a** Problem 1—Action Interpolation. **b** Problem 2—Transfer to novel actions. **c** Problem 3—Transfer to novel shapes (Color figure online)

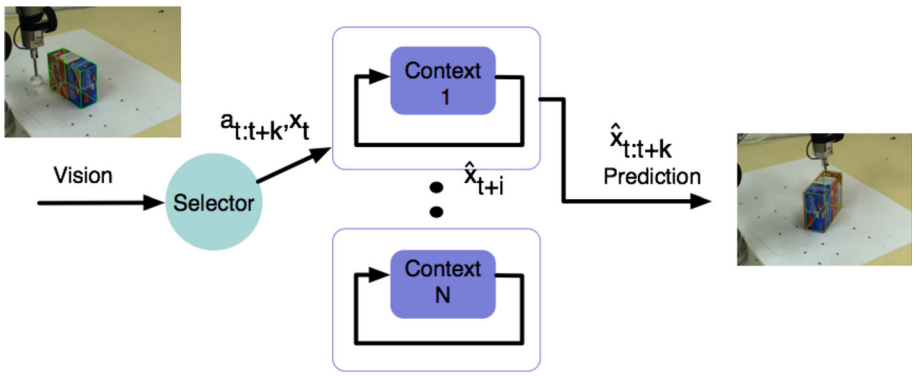



**Fig. 2** A modular prediction scheme for solving problem P1. Visual object identification selects a context/predictor, and gives it the object pose and intended finger trajectory as initial input. The chosen predictor takes the current system state $x_t$ and the planned manipulator trajectory $a_{t:t+k}$. The first prediction $\hat{x}_{t+1}$ is fed back on itself to produce a multi-step prediction $\hat{x}_{t:t+k}$

from the available data. Thus, even for a classical mechanics approach, the problem involves learning. Alternatively, generalisation across actions is feasible using semi- or non-parametric machine learning. This is because the experiences span the test case: there aren't any exactly similar actions, but there are many with similar features. Hence, this problem involves *action interpolation*.

### 3.2 Action transfer (P2)

Figure 1b depicts a harder problem since the test action (bottom row) now sits outside the range of training actions. Hence, this problem is known as *action transfer*. Turning the object around, since it is not symmetric, means that the effects of actions are quite different than before. For example, pushing the top of the L-shaped object will no longer induce it to tip over. This is because the horizontal flap cannot pass through the table: it provides a *kinematic constraint* on the motion of the object. This makes action transfer problems challenging for tabula rasa machine learning. Such problems should, however, be no more challenging for rigid body simulation than problem P1, since once an object's parameters are estimated, the rigid body simulator can produce predictions for any action.

### 3.3 Shape transfer (P3)

Finally, Fig. 1c requires generalising predictions about action effects to novel shapes. The training data consists of pushes of two objects of different shape. The test action is a push of an object of novel shape. This is a challenge for learning because small changes in object shape can lead to large changes in behaviour. The problem is also challenging for an approach that uses a tuned rigid body simulator, since estimation of mass and frictional coefficients for the test object must be based on the estimates made from the training data, and will thus be sensitive to estimation errors.

Problems P2 and P3 arise from quite different kinds of variation, but in fact are quite similar. Both require learning of the possible motion types at contacts. By learning models of contact behaviour, rather than whole object motion, both problems can be solved in essentially the same way.

### 3.4 The case for modular prediction learning

In this paper we pursue a learning approach to prediction. We could try to learn a single predictor, applicable to a wide range of objects and contexts. This is hard, however, because a great deal must be captured in a single learner. Instead, we employ a modular prediction scheme, inspired by MOSAIC (Haruno et al. 2001) and other models (Demiris and Johnson 2003). Modular means that the overall prediction engine consists of many context specific predictors (Fig. 2), where a context is an object, or an object-environment combination. The first advantage of this is that it can be easier to solve many simple learning problems than one complex learning problem. Second, unobservable parameters (frictional coefficients, mass, mass distribution) need not be modelled explicitly, but are instead captured implicitly by being associated with a particular context. Whereas MOSAIC couples control and prediction, but avoids real objects (working with simulated mass spring systems). Our work focuses on pure prediction, but for real objects. Our modular prediction scheme uses vision to distinguish the context, by identifying an object shape from a library. We also show how to decompose the prediction models into recombinable components, by factoring them. The models are factored by the mechanical contacts. This factoring allows transfer learning.[1] Having explained our overall scheme, we now turn to the mathematical details of how to model robot-object-environment interactions. This will lead, in turn, to posing the three prediction problems formally.

[1] This factoring could also be referred to as modular, in the sense that the factors can be combined and replicated. To avoid confusion, however, we use module only to refer to a context specific predictor. We use factoring to refer to elements in the recombinable scheme.

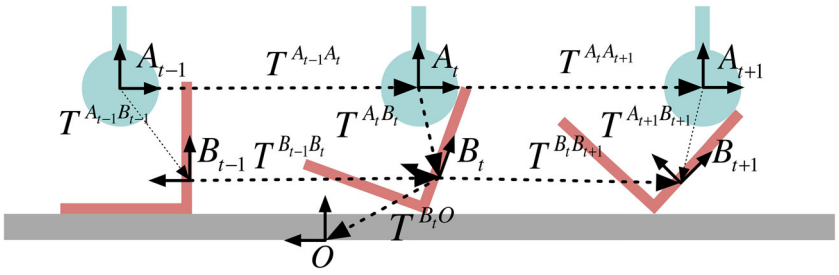



**Fig. 3** 2D projection at time $t$ of a robotic finger with frame $A_t$ at time $t$, an object with frame $B_t$, and a ground plane with constant frame $O$. The system can be adequately described using six rigid body transformations. We show all the transformations referred to in the paper, marking the six transformations we use for framing our general rigid body prediction problem as *bold dotted lines*, with the others shown as *non-bold dotted lines*. The transforms $T^{A_t,B_t}$, $T^{B_t,O}$, $T^{A_t,A_{t+1}}$ are used for the reduced quasi-static formulation

## 4 Encoding rigid body kinematics

We now set up the required notation. Without loss of generality, we explain this using an example from our application domain (Fig. 3). Three reference frames $A$, $B$ and $O$, sit in a 3-dimensional Cartesian space. Frame $A$ is attached to a robot finger, which pushes an object with frame $B$, which in turn is placed on a table top with frame $O$.[2] While frame $O$ is fixed, $A$ and $B$ change in time and are observed at discrete time steps $..., t-1, t, t+1, ....$ Frame $X$ at time step $t$ is denoted $X_t$, and the rigid body transformation from a frame $X$ to a frame $Y$ is denoted by $T^{X,Y}$.

From classical mechanics, we know that in order to predict the change in state of a rigid body it is sufficient to know its mass, velocity, and a net force applied to the body. We do not assume any knowledge of the mass and applied forces, learning only from object trajectories.[3] We can, however, use the motion of a body over time to encode acceleration—an effect of the applied net force. We therefore use rigid body transformations, $T^{X,Y}$, of the interacting bodies through time (Fig. 3). Given the additional assumption that the net force and the body mass are constant, two subsequent rigid body transformations, $T^{B_{t-1},B_t}$ and $T^{B_t,B_{t+1}}$, give a complete description of the state of some body B (here the object) at time step $t$, in the absence of the other bodies. Adding the transformation $T^{B_t,O}$, to give a triple of transformations, thus provides a complete description of the state of body B in the fixed frame $O$ (the stationary elements of the environment). Similarly, a second triple of transformations, $T^{A_t,O}$, $T^{A_{t-1},A_t}$ and $T^{A_t,A_{t+1}}$, provides such a description for some other body (here the finger) with frame $A$. The state of these two interacting bodies, with frames $A$, $B$ and the fixed environment $O$, can thus be adequately described by these six transformations.

In fact, we replace transformation $T^{A_t,O}$ by relative transformation $T^{A_t,B_t}$. This explicitly captures the spatial relationship, and thus any contacts, between $A$ (finger) and $B$ (object). This gives us a representation consisting of the set of six transformations marked in bold dotted lines in Fig. 3. The general prediction problem is to predict the motion, $T^{B_t,B_{t+1}}$, of the object $B$ given these five transformations. In fact, in the experimental work described in this paper, we reduce the problem to a quasi-static one, relying only on transformations $T^{A_t,A_t+1}$, $T^{B_t,O}$, and $T^{A_t,B_t}$. This dimensionality reduction makes learning easier.[4] Next, before defining the problem formally, we need to think briefly about how best to store these transformations.

Specifically, since we are interested in learning, we need to express this set of transformations in a way that supports generalised predictions. We make extensive use of transformations relative to the frame of the object about which predictions are made. Thus all transformations for learning are expressed in a frame attached to the body of the object, e.g. $T_{body}^{X_t,X_{t+1}}$. At prediction time these transformations are converted into a general inertial frame located in the world, thus becoming for example $T_{in}^{X_t,X_{t+1}}$. This technique is critical to generalisation across inertial frames. In the rest of the paper we will retain subscripts $in$, but suppress subscripts $body$. Thus all transformations denoted $T^{X,Y}$ are transformations in the body frame $X$, related to the equivalent transform in some inertial frame using a similarity transform:

$$T^{X,Y} \equiv T_{body}^{X,Y} = (T^{I,X})^{-1} T_{in}^{X,Y} T^{I,X} \tag{1}$$

## 5 Formal statement: learning to predict

We now have the basics required to formally describe the one-step and then the multi-step prediction problem in such

[2] Although it is an abuse of notation, we use $A$, $B$ and $O$ to denote both the frame and the bodies to which they attach.

[3] We could include forces, and sense them with a force/torque sensor. This is future work.

[4] It may be that other representations contain enough information to predict. To avoid the quasi-static assumption it may be that finding a subspace in which the sets of transformation typically lie is a promising route.

a way that they become problems of learning to predict, and we can effectively tackle problem P1 (Action Interpolation).

### 5.1 One step prediction

The one step prediction problem is formulated as follows. Given observations of the recent poses of the finger and object, and the planned motion of the finger, $T^{A_t,A_{t+1}}$, predict the resulting immediate motion of the object, $T^{B_t,B_{t+1}}$. This is a problem of finding a function $f$:

$$f : T^{A_t,B_t}, T^{B_t,O}, T^{A_{t-1},A_t}, T^{B_{t-1},B_t}, T^{A_t,A_{t+1}} \longrightarrow T^{B_t,B_{t+1}} \quad (2)$$

The function $f$ is capable of describing the effects of interactions between rigid bodies $A$ and $B$, provided that their physical properties and net forces are constant in time,[5] in the limit of infinitesimally small time steps. Furthermore, it can be approximately learned from observations, for some small fixed time interval $\Delta t$ between time steps.

If robotic manipulations are performed slowly we can assume quasi-static conditions, and ignore all frames at time $t-1$. This conveniently reduces the dimensionality of the problem, giving a simplified function $f_{qs}$:

$$f_{qs} : T^{A_t,B_t}, T^{B_t,O}, T^{A_t,A_{t+1}} \longrightarrow T^{B_t,B_{t+1}} \quad (3)$$

It is this quasi-static formulation that is used to train and test the learning methods in this paper.

### 5.2 Multi-step prediction

Having stated the one-step prediction problem it is possible to solve the multi-step prediction problem.[6] Given a predictor (either $f$ or $f_{qs}$), the initial states of the finger, $T^{A_1,O}$, and object, $T^{B_1,O}$, and knowing the trajectory of the finger $A_1, \ldots A_T$ over $T$ time steps, one can predict the complete trajectory of the object $B_1, \ldots B_T$, by simply iterating the predictions obtained from $f_{qs}$. So, the output of the predictor at time $t$ is used as the input to the predictor for the next time step (Fig. 2). While this is a well known approach, it is difficult to produce a predictor that will behave well over many time steps. Over time all predictors will diverge from reality. Thus, an empirical question is whether, for a particular domain and prediction scheme, predictions are reasonably close to reality, over a suitable number of steps. It is this multi-step prediction problem that is solved in this paper.

### 5.3 Learning to predict as regression

In principle it is straightforward to acquire a predictor, $f$ or $f_{qs}$, by learning it from data. Given sufficient experience of object and finger trajectories, we can perform a nonparametric regression analysis, by taking $T^{A_t,B_t}, T^{B_t,O}, T^{A_t,A_{t+1}}$ as independent variables, and $T^{B_t,B_{t+1}}$ as the dependent variable. Nonetheless, a powerful regression technique is needed, since the domain of $f_{qs}$ has 18 dimensions or more, depending on the parameterisation of motion.

### 5.4 Learning to predict as density estimation

As an alternative to learning the mapping (3) by regression, we can recast $f_{qs}$ as a conditional probability density (CPD) $p_{qs}$ over possible object motions $T^{B_t,B_{t+1}}$ (Kopicki et al. 2009):

$$p_{qs}(T^{B_t,B_{t+1}} | T^{A_t,B_t}, T^{B_t,O}, T^{A_t,A_{t+1}}) \quad (4)$$

The learning problem is then posed as one of density estimation. This permits modelling the probabilities of many possible outcomes.

Either the regression or density estimation formulation can be used in a modular scheme. In this case a separate module is learned for each combination of agent, object, and environment. Each module interpolates over actions for its context, and thus the overall system solves problem P1. We now identify the additional information needed to solve problems P2 and P3.

## 6 Transfer learning

### 6.1 The need for contact models

The input domains of $f$, $f_{qs}$, and $p_{qs}$ are insufficient to pose problems P2 (Action Transfer) or P3 (Shape Transfer). This is because they only capture the *global* relations between objects. To properly pose transfer learning, the input domain must capture all the *local* contact relations between the object and its surroundings. To see why consider Fig. 4. On the left is a training example. On the right is a test case, where the object is wider. Given the same placement of the frames on object and agent, and the same finger motion, the predicted behaviour using Eq. (3) will be the same as for the training example. This is wrong. For the correct prediction to be transferred, additional information is needed about the contact between $A$ and $B$. This can be captured by attaching additional frames to $A$ and $B$ (Fig. 5). In general, an object

[5] A dynamic formulation could explicitly incorporate net forces into the domain and codomain of (2).

[6] In this paper we study the multi-step problem in all our experiments. While the single step problem has been studied for other time series its utility for manipulation planning is low. Second, given fine grained single step predictions it is very hard to distinguish prediction quality over a single step.

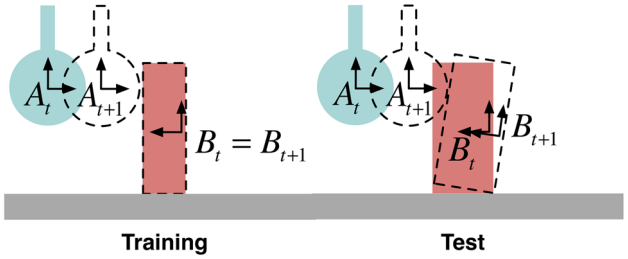



**Fig. 4** Two scenes (*left* and *right*), each with an object on a tabletop. Only the shape of object $B$ differs between the scenes. Yet, when finger A moves, as shown by the *dashed* outline at time $t+1$, the resulting transformation of $B$ will be quite different

has multiple contacts with the robot and the environment. Each of these contacts provides a kinematic constraint on the object's motion, and thus each one should be modelled. Rigid body simulators use just such contact information.

### 6.2 Modelling contacts and near contacts

We use a pair of local frames to encode each contact or near contact. Each pair encodes a transformation between part of the object $B$ and another body. To distinguish these local frame pairs from what has gone before we henceforth refer to the main frame attached to a body (defined in Sect. 4) as that body's global frame. We define the local frame pairs as follows. Consider Fig. 5. We first define a pair of local frames capturing the finger-object contact as $A_t^L$ and $B_t^L$ (centre panel). These are spatially dynamic, i.e. at any time $t$ they are located at the points of closest proximity on the finger and object respectively. We define the *agent-object contact* information as the transformations $T^{A_t^L, A_{t+1}^L}$ and $T^{A_t^L, B_t^L}$.

We also define local frame pairs, to model object-environment contacts. One frame is attached to some point on the object ($B_t^{Sk}$), and one is attached to the nearest point in the environment $E_t^{Sk}$. The object frames are attached to a few points with distinctive local curvatures on the training object, or failing that at regular intervals on the constant curvature surface. Thus the environment frame within each pair is spatially dynamic, changing its position as the object moves. If $N$ points on the object are chosen for modelling there will be $N$ pairs of local frames $B_t^{Sk}$ and $E_t^{Sk}$ to capture the object-environment contacts at time $t$, where $(k = 1 \ldots N)$ (Fig. 5 right panel). Using these frame pairs, we then define the *object-environment contact* information as the set of transformations $T^{E_t^{Sk}, B_t^{Sk}}$ for $k = 1 \ldots N$.

This section described the information required to allow transfer learning. We refer to this as *contact* information, even though it also includes information on surfaces not in, but close to, contact.

### 6.3 Learning a contact model

Given observations of moving surfaces in contact, and near contact, it is possible to learn contact models. In this paper, during training, two contact models are learned: an agent-object contact model, and an object-environment contact model. The object-environment contact model is created by pooling data from frames located at several points on the object. Thus, a learned model of contact behaviour is created from many contact examples. This contrasts with the analytic approach used by rigid body simulators. An extension would be to condition this model on other variables, for example information on local surface properties, such as shape or texture. We hypothesize that both data pooling and conditioning will be important elements in improving the transfer of predictions.

### 6.4 Predicting with contact models

During prediction the learned contact models are applied to the test case. This requires mapping the various reference frames from the training examples to the test examples. For problems P1 and P2 this is trivial, since the objects are the same. For problem P3 the placement can be made in several ways. In our implementation we used heuristic rules. The

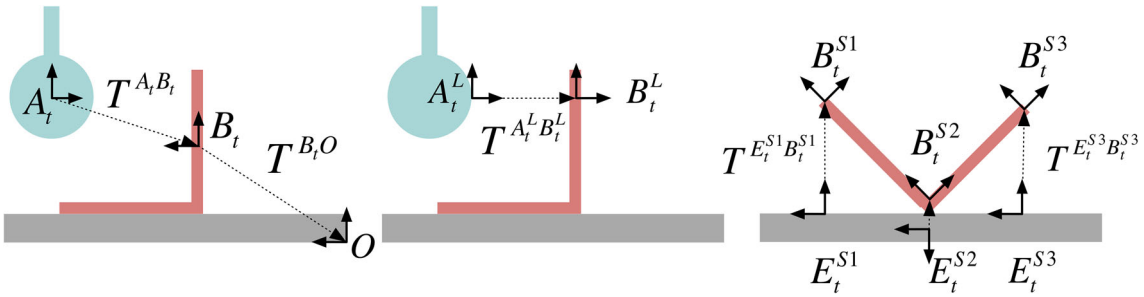



**Fig. 5** Three types of information useful in prediction problems. *Left* (G—global) global frames of reference for the robot, object and the world. *Centre* (A—agent) local frames of reference on the robot finger and the closest point on the object. *Right* (E—environment) local frames of reference on the object and the closest points on surfaces in the environment

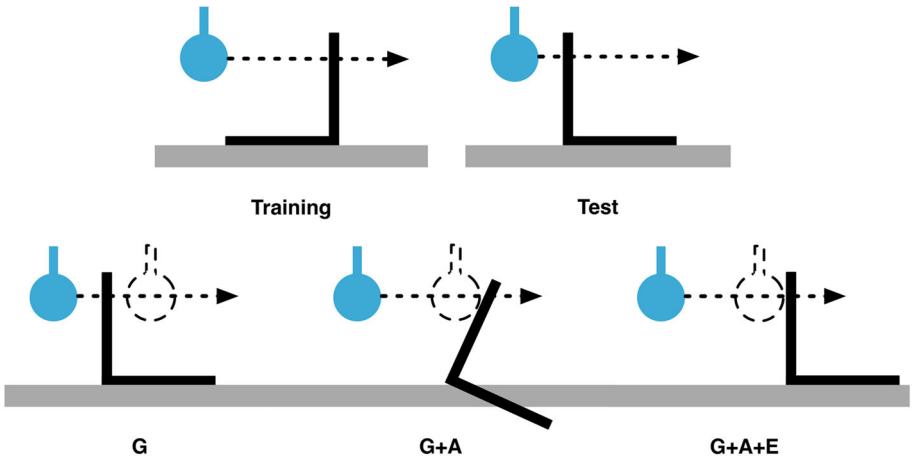



**Fig. 6** Information for Action Transfer: an L-shaped object is pushed by a finger. Various predictors are trained solely on forward pushes (*top left*), but tested on backwards pushes (*top right*). The *top panels* show a training and a test push, whereas the bottom panels show predictions given different information (G, G+A and G+A+E) for the test push

global frame was attached to the most central point lying within the object. The agent-object frames were automatically attached, each step, to the object location closest to the finger. Finally, $N$ object-environment frames, $B_t^{Sk}$, were attached to points on the object surface with curvature sufficiently similar to that in the training data which contributed to the contact model. This heuristic strategy was empirically determined to be robust. Because of data pooling during training, each object-environment expert is a copy of a single object-environment model. Thus several, identical, contact experts are created on the new object. Other ways to automate the placement strategy for P3 are possible.

Finally, we note that we have motivated these contact experts as enabling shape transfer, but they are equally applicable to action transfer. The top row of Fig. 6 shows a training and a test case for problem P2 (action transfer). The prediction for the test case requires encoding of the kinematic constraint imposed by the contact between the base of the L-shaped flap and the table. This constraint also existed in the training push, but was not significant since the flap could rotate on its corner.

### 6.5 Hypothesized benefits of contact modelling

We can now consider the effects that different sets of information might have. We shall refer to a predictor that uses only the global frames, $A$ and $B$, as having global information (G). We can add agent-object contact information (G+A), and object-environment contact information (G+A+E). Now consider the possible predictions for the test case (Fig. 6 bottom row). A predictor using G will predict that the object will not move. A predictor using G+A has information, from the training case, that the object surface will move with the finger, so that the finger will not pass through it. But this information is also capable of predicting that the object rotates about the corner and into the table, since it doesn't model the object-environment contact. A predictor using G+A+E will have information about the effect of the contact between the base of the flap and the table, and so should avoid predicting a rotation into the table. One point is critical, this analysis only concerns what the information allows. Performance will depend on a learner's ability to utilise it.

### 6.6 Reformulating prediction with contact information

We can simply extend the prediction formulations to incorporate contact information. For regression we simply enlarge the domain of function $f$ in Eq. (3):

$$f'_{qs} : T^{A_t,B_t}, T^{B_t,O}, T^{A_t,A_{t+1}}, T^{A_t^L,B_t^L} \{, T^{E_t^{Sk},B_t^{Sk}}\}_{k=1\ldots N} \longrightarrow T^{B_t,B_{t+1}} \quad (5)$$

Recall that, at prediction time, we will need $N$ copies of the object-environment contact information, which was pooled at the learning stage. Hence, we use $k$ to index over these copies. Unfortunately, because the dimensionality of the domain of $f'_{qs}$ grows with the number of environment contacts, $N$, the difficulty of learning the mapping $f'_{qs}$ rapidly increases as environment contacts are added.

The conditional probability density (CPD) $p_{qs}$ over possible object motions $T^{B_t,B_{t+1}}$ (Kopicki et al. 2009) is augmented as follows:

$$p_{qs}(T^{B_t,B_{t+1}} | T^{A_t,B_t}, T^{B_t,O}, T^{A_t,A_{t+1}}, T^{A_t^L,B_t^L} \{, T^{E_t^{Sk},B_t^{Sk}}\}_{k=1\ldots N}) \quad (6)$$

Again, the dimensionality of the conditioning variables makes density estimation hard as the number of contacts grows. One way around this, in the density estimation case, is to factorize the density in a way that reflects the contact structure. We consider this in the next section.

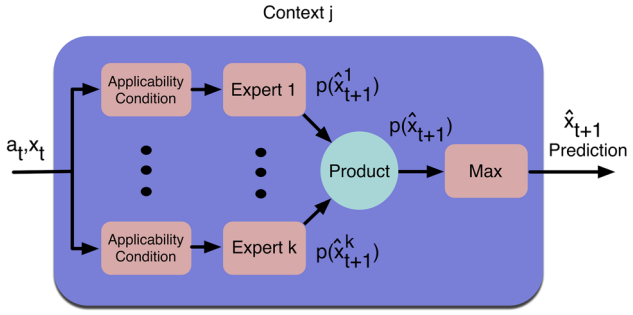



**Fig. 7** A Dynamic Product of Experts. This gives the structure of a factorised predictor for a single context, as depicted in the modular learner in Fig. 2. Each expert in the product has an applicability condition, which determines whether it contributes to the product. The applicable predictors combine densities over predictions to produce an overall density. This is optimised to produce a specific prediction

## 7 Factorised density estimation

Both formulations give learning problems that increase in difficulty as further contacts are added. One question is whether either formulation can be recast, so as to take advantage of the natural problem structure. This section presents one such scheme for the density estimation (or CPD) formulation, based on a product of experts (Fig. 7).

Specifically, the CPD formulation allows us to factorise the density, and approximate $p_{qs}$, by making a conditional independence assumption. The unfactored CPD formulation gives a density over possible one step motions of the object. We can factorise this by breaking up the conditioning variables into groups, according to the contacts. This reflects the notion that the behaviour at each contact is independent of the other contacts. Each component of the product is an expert, which encodes the likely object motions given a single kinematic constraint. The product will be maximised by a motion that best satisfies all the constraints simultaneously.

The computational advantage is that, since the component densities factorise the conditioning variables of $p_{qs}$, the overall predictor works better with a high dimensional input space. Furthermore, the subset of experts used in the product can be selected dynamically, depending on the current set of contacts. For some normalisation constant $C$ we propose the following factorisation:

$$p_{qs} \approx C\ p_{global}\ p_{agent} \prod_{k=1\ldots N} p_{env,k} \tag{7}$$

where

$$p_{global} \equiv p_{global}(T^{B_t,B_{t+1}} | T^{A_t,A_{t+1}}, T^{A_t,B_t}, T^{B_t,O}) \tag{8a}$$

$$p_{agent} \equiv p_{agent}(T^{B_t^L,B_{t+1}^L} | T^{A_t^L,A_{t+1}^L}, T^{A_t^L,B_t^L}) \tag{8b}$$

$$p_{env,k} \equiv p_{env,k}(T^{B_t^{Sk},B_{t+1}^{Sk}} | T^{E_t^{Sk},B_t^{Sk}}) \tag{8c}$$

denote the *global*, *agent-object*, and $k^{th}$ *object environment* density factors, respectively (Kopicki et al. 2009; Kopicki 2010). Recall that each of the experts $p_{env,k}$ is a replica of a single common contact model. The one step prediction problem is now defined as finding the transformation $\widetilde{T}_{in}^{B_t,B_{t+1}}$, expressed in some inertial frame, which maximises the product of densities (7):

$$\widetilde{T}_{in}^{B_t,B_{t+1}} = \underset{T_{in}^{B_t,B_{t+1}}}{\text{argmax}} \left\{ p_{global}\ p_{agent} \prod_{k=1\ldots N} p_{env,k} \right\} \tag{9}$$

where similarity transforms, as described in Sect. 4, must be used to evaluate $p_{global}$, $p_{agent}$, and the $N$ environment factors $p_{env,k}$, for a given $T_{in}^{B_t,B_{t+1}}$.

The key property here is that the global, agent, and environment densities encode different information as to which rigid body transformations are feasible. By taking the product of these densities, only transformations which are feasible in all factors' frames will have high probability in the resulting combined distribution. In addition, we make this product dynamic in the number of object-environment factors. Once the object surface is above some threshold distance from the environment surface its predictor switches off, and when it is close enough it switches on again. This enables us to keep only relevant predictors in the product at any one time—improving prediction quality and efficiency.

In summary, we have now described two main formulations (regression and density estimation) able to incorporate varying amounts of information (G, G+A, G+A+E). We have also presented a reformulation of density estimation that factorises the prediction problem, given information G+A or G+A+E, into a product of experts. Which information and problem formulation should be combined to provide the best prediction framework? Is the factorised problem better able to exploit the additional information than the unfactorised version? These questions can only be answered using specific regression and density estimation algorithms. Having completed our problem formulation we therefore now turn to the details of the implementations for each framework.

## 8 Implementation

The implementations are indexed by the algorithms used, and the information employed. For the function approximation formulation we used Locally Weighted Projection Regression (LWPR). For the unfactored density estimation formulation we used a variant of Kernel Density Estimation (KDE), and for the factored density estimation formulation we also used KDE, but denote it KDEF where -F denotes the use of factorisation. In addition, each algorithm: LWPR, KDE, KDEF, was implemented with differing amounts of

**Table 1** Input & output dimensionality

| Predictor | Input dim. | | | Output dim. |
|---|---|---|---|---|
| LWPR-G (e) | 18 | | | 6 |
| LWPR-GA (e) | 24 | | | 6 |
| LWPR-GAE (e) | 24 + N*6 | | | 6 |
| | Global | Agent | Env | |
| KDEF (e) | 18 | 12 | 6 | 6 |
| KDEF (q) | 21 | 14 | 7 | 7 |
| KDEF (v) | 21 | 14 | 7 | 7 |

There are $N$ "environment contacts", so there are $N$ environment experts. The abbreviations refer to the parameterisation of rigid body transforms: (e) Euler, (q) Gauss quaternion, (v) Von Mises–Fisher quaternion

input information. We denote these G (Global), GA (Global and Agent) and GAE (Global and Agent and Environment), as described previously. All the implementations depend on the parameterisation of rigid-body transformations chosen. In this paper we tested two parameterisations of orientation: Euler angles and quaternions (see e.g. (Murray et al. 1994)). We also employed two different densities for the quaternion parameterisation: Gaussian and von-Mises Fisher.

### 8.1 Regression method

LWPR (Vijayakumar et al. 2005) is a powerful method, applied widely in robotics, for estimating the mapping described by Eq. (3). The regression scheme was implemented using the LWPR software library (Klanke et al. 2008). LWPR was chosen because it employs an incremental learning algorithm that can handle a large number of input dimensions. After initial experimentation, LWPR was run using the Euler angle parameterisation. The dimensions of the input and output spaces of each LWPR predictor are summarised in Table 1.

### 8.2 Kernel density method

A variant of Kernel Density Estimation (KDE) (Scott and Sain 2004) is used to approximate the conditional densities employed in the product in Eq. (9). This requires that we encode the rigid body transformations as parameter vectors. To that end, and for the sake of compactness, we introduce some additional notation. First $T^{\mathbf{x}_f}$ is used to denote the set of conditioning transformations for factor $f \in \{G, A, (E,1)\ldots(E,N)\}$, and $T^{\mathbf{y}_f}$ for the corresponding conditioned transformation. $\mathbf{x}_f$ and $\mathbf{y}_f$ are then simply the corresponding parameter vectors for a given parameterisation. Given this, the global factor, for example, can be referred to in three equivalent ways:

$$p_G(T^{B_t,B_{t+1}}|T^{A_t,A_{t+1}}, T^{A_t,B_t}, T^{B_t,O}) \tag{10}$$
$$\equiv p_G(T^{\mathbf{y}_G}|T^{\mathbf{x}_G}) \tag{11}$$
$$\equiv p_G(\mathbf{y}_G|\mathbf{x}_G) \tag{12}$$

Finally, we define the parameter vectors for the unfactored density estimation problem Eq. (6) as the concatenation of the vectors for each factor, so that

$$\mathbf{x} = \langle \mathbf{x}_G, \mathbf{x}_A, \{\mathbf{x}_{E,k}\}_{k=1\ldots N}\rangle \tag{13}$$
$$\mathbf{y} = \langle \mathbf{y}_G, \mathbf{y}_A, \{\mathbf{y}_{E,k}\}_{k=1\ldots N}\rangle \tag{14}$$

So as to capture the conditional probability densities (CPD) over $\mathbf{y}_f$, for all the different values of $\mathbf{x}_f$, we simply perform kernel density estimation for their joint density, and then index by the specific $\mathbf{x}^t$ at prediction time to give $p_f(\mathbf{y}_f^t|\mathbf{x}_f^t)$. For factor $f$ the joint density kernel estimate is:

$$p_f(\mathbf{y}_f, \mathbf{x}_f) \propto \sum_{j=1\ldots M} K_{\mathbf{H}^{\mathbf{x}_f}}(\mathbf{x}_f - \hat{\mathbf{x}}_f^j) K_{\mathbf{H}^{\mathbf{y}_f}}(\mathbf{y}_f - \hat{\mathbf{y}}_f^j) \tag{15}$$

where the bandwidth matrices $\mathbf{H}^{\mathbf{x}_f}$ and $\mathbf{H}^{\mathbf{y}_f}$ are diagonal, so that $\mathbf{H}_{ii}^{\mathbf{x}_f} = \theta \mathbf{h}_i^{\mathbf{x}_f}$. Vectors $\mathbf{h}^{\mathbf{x}_f}$ and $\mathbf{h}^{\mathbf{y}_f}$ are estimated from training samples using Silverman's "multivariate rule-of-thumb" (Scott and Sain 2004). The additional scaling parameter $\theta \in \mathbb{R}$ is estimated by model selection (see Sect. 9.2). Note that $\mathbf{H}^{\mathbf{x}_f}$ and $\mathbf{H}^{\mathbf{y}_f}$ depend on the factor $f$ being estimated. Given that they are diagonal, and suppressing $f$ and $\theta$ for compactness, each kernel function $K()$ in Eq. (15) can thus be written:

$$K_{\mathbf{H}^{\mathbf{x}}}(\mathbf{x} - \hat{\mathbf{x}}) = \exp\left[-\frac{1}{2} d(\mathbf{x}, \hat{\mathbf{x}}, \mathbf{h}^{\mathbf{x}})\right] \tag{16}$$

where $d()$ is a *distance function* that determines the kernel type. We employed Gaussian kernels for both Euler and quaternion representations, and additionally Gaussian+Von Mises Fisher kernels for the case of quaternions. For a *Gaussian kernel*:

$$d_{\mathcal{N}}(\mathbf{x}, \hat{\mathbf{x}}, \mathbf{h}) = (\mathbf{x} - \hat{\mathbf{x}})^{\mathrm{T}} \mathbf{H}^{-1} (\mathbf{x} - \hat{\mathbf{x}}) \tag{17}$$

For a product of a Gaussian kernel and *von Mises–Fisher* kernel:

$$d_{\mathcal{N}\mathrm{V}}(\mathbf{x}, \hat{\mathbf{x}}, \mathbf{h}) = d_{\mathcal{N}}(\mathbf{p}, \hat{\mathbf{p}}\mathbf{h}^{(p)}) + d_{\mathrm{V}}(\mathbf{q}, \hat{\mathbf{q}}, h^{(q)}) \tag{18}$$

where $\mathbf{h} = \left[\mathbf{h}^{(p)}; h^{(q)}\right]$, $\mathbf{h}^{(p)} \in \mathbb{R}^3$, $h^{(q)} \in \mathbb{R}$, and [see e.g. (Abramowitz and Stegun 1965)]:

$$d_{\mathrm{V}}(\mathbf{q}, \hat{\mathbf{q}}, h^{(q)}) = 2h^{(q)}\left(1 - \left|\mathbf{q} \cdot \hat{\mathbf{q}}\right|\right) \tag{19}$$

where $\mathbf{q} \cdot \hat{\mathbf{q}}$ is the quaternion dot product, and taking the absolute value fixes the double cover problem. This Von

Mises–Fisher kernel is an approximation (up to a multiplicative constant (Detry 2010)) of the von Mises–Fisher distribution.

The learning algorithm, for a single context, for factored KDE is now straightforward, given a set of $S$ training sequences $\{(\hat{\mathbf{x}}^{1:\tau}, \hat{\mathbf{y}}^{1:\tau})_s\}_{s=1\ldots S}$, each of length $\tau$. The kernel centres are simply stored in a set $\mathcal{K} = \{(\hat{\mathbf{x}}_f^j, \hat{\mathbf{y}}_f^j)\,\forall f \in F\}_{j=1\ldots M}$, where $\hat{\mathbf{x}}_f^j$ denotes the $j^{th}$ kernel centre, and $M = \tau \times S$. A kernel bandwidth is computed for $\mathbf{x}_f$ and $\mathbf{y}_f$ for each factor $f$, and stored in a set $\mathcal{H} = \{(\mathbf{H}^{\mathbf{x}_f}, \mathbf{H}^{\mathbf{y}_f})\}_{f\in F}$. The training procedure for KDE is identical, but with only one factor. The dimensionality of the input ($\mathbf{x}$) and output ($\mathbf{y}$) spaces for different KDEF predictors is summarised in Table 1. The equivalent unfactored KDE space is obtained by summing over the factors of the equivalent KDEF predictor. Each object is trained as a separate context to form a modular predictor.

### 8.3 Prediction

Single step prediction for LWPR is straightforward, but single step prediction for KDE involves optimisation of the likelihood of the prediction, and for KDEF this is non-trivial. We describe that here. Following Eq. (9), the single step prediction problem can be defined as finding the transformation $T^{\mathbf{y}^*}$, parameterised by $\mathbf{y}^*$, which maximises the product of conditional densities (7) given query $\mathbf{x}$, i.e.:

$$\mathbf{y}^* = \underset{\mathbf{y}}{\text{argmax}}\; p_{qs}(\mathbf{y}|\mathbf{x}) \tag{20}$$

The one-step prediction algorithm is given in Fig. 8. First, for the input vector $\mathbf{x}$, the conditional density over $\mathbf{y}$ must be obtained for each factor $f$. This is achieved by evaluation of each kernel $K_{\mathbf{H}^{\mathbf{x}_f}}(\mathbf{x}_f - \hat{\mathbf{x}}_f^j)$, from $j = 1 : M$ in Eq (15), to give a weight $w_{f,j}$. The vector of normalised weights $\mathbf{w_f}$ forms a distribution over the kernels in $\mathbf{y}_f$. $\mathbf{w_f}$ is computed for every factor $f$. For efficiency we only consider the $r$ kernels with the highest weights $w_{f,j}$ at the query point $\mathbf{x}_f$.

**one-step-prediction-KDEF**$(\mathbf{x}) \rightarrow \mathbf{y}^*$
$F = (G, A, (E,1)\ldots(E,N))$
**for** $f \in F$ **do**
  **for** $j = 1$ to $M$ **do**
    $w_{f,j} = K_{\mathbf{H}^{\mathbf{x}_f}}(\mathbf{x}_f - \hat{\mathbf{x}}_f^j)$
  **end for**
  $\mathbf{w_f} = \text{normalisation}(w_{f,1}\ldots w_{f,M})$
**end for**
**for** $i = 1$ to $\beta$ **do**
  randomly sample a factor $f \in F$
  sample $j$ from distribution $\mathbf{w_f}$
  $\mathcal{Y}_i = \tilde{\mathbf{y}}$ sampled from density with mean $\hat{\mathbf{y}}_f^j$ and bandwidth $\mathbf{H}^{\mathbf{y}_f}$
**end for**
maximise Eq.10 with $Y^* = \text{differential-evolution}(\mathcal{Y},\mathcal{K})$

**Fig. 8** One-step prediction for the KDEF method

**Table 2** Algorithm-information variants

| | Information | | |
|---|---|---|---|
| Predictor | G | G+A | G+A+E |
| LWPR | LWPR-G | LWPR-GA | LWPR-GAE |
| KDE | KDE-G | KDE-GA | KDE-GAE |
| KDEF | KDEF-G | KDEF-GA | KDEF-GAE |
| PhysX | n/a | n/a | n/a |

A initial population of solutions is then generated by sampling. First, a factor $f$ is sampled randomly. Then a kernel $j$ is sampled by drawing $j$ according to the distribution $\mathbf{w}_f$. Finally, a candidate $\tilde{\mathbf{y}}$ is sampled from the $j$th kernel with centre $\hat{\mathbf{y}}_f^j$. This sampling procedure is run $\beta$ times to create the initial set of candidate solutions.

This initial solution set is then refined by stochastic optimisation with respect to $p_{qs}$. To achieve this, similarity transforms must be used to evaluate the likelihood of each $\mathbf{y}$ according to each factor $f$. Any optimisation routine could be applied, but we used differential evolution (DE) (Storn and Price 1997).[7] DE is particularly simple to tune, since it has two meta-parameters: *crossover probability* $\alpha$ and *population size* $\beta$. If the number of generations is fixed, the total run time scales linearly with the number of factors, their dimensionality, and the number of samples. Optionally, the entire maximisation procedure is stopped when no further significant improvement is observed.

In order to solve the multi-step prediction problem, the prediction from the one step prediction algorithm $\mathbf{y}^t$ is fed back as the relevant part of the conditioning parameter vector $\mathbf{x}^{t+1}$. In this way long prediction sequences can be generated for all the algorithms. For efficiency, no learning or prediction was performed in a given trial until the initial contact was made between the robot and object.

## 9 Experimental study

### 9.1 Overview of experiments

We conducted three experiments, one for each problem in Sect. 3: P1 (action interpolation), P2 (action transfer), and P3 (shape transfer). Each experiment tests all combinations of information (G,A,E) and learning algorithm (LWPR, KDE, KDEF) (see Table 2). The structure of these experiments reflects the structure of our hypotheses: H1-H3. A summary of experimental results are available in video form.[8]

[7] Note that one cannot use the mean-shift algorithm (Cheng 1995) due to the product involved in (20).

[8] https://youtu.be/bSp9y4S0sQ4.

#### 9.1.1 Experiment P1 (action interpolation)

This tests whether learning methods successfully interpolate on the action space. Results were obtained with both real objects, and in simulation. The real object experiment tests hypothesis **H1***: a modular learning approach can outperform physics engines for prediction of rigid body motion.* Each learning method was used to learn a context/object specific predictor, and the results were compared to the predictions of a physics simulator that had also been tuned to each object in a modular manner. We also compared the effects of different parameterisations of rigid body transformations. The simulated experiment tests whether interpolated predictions are also good when the object is manipulated on a non-planar surface.

#### 9.1.2 Experiment P2 (action transfer)

This tests hypothesis **H2***: learned predictions can be transferred to novel actions.* The learners were trained with a reduced action set on a non-symmetric object, and then tested on that object with novel actions. We performed this in simulation and with a real object.

#### 9.1.3 Experiment P3 (shape transfer)

This tests hypothesis **H3***: learned predictions can be transferred to novel shapes.* Learners were trained on one or more objects, and tested on an object of different shape. Transfer was tested both in simulation and with real objects.

### 9.2 Experimental setup

In each trial the object was placed in a fixed location. Random pushes were generated as follows. A target point on the object surface was selected from a uniform distribution over the interval of the vertical plane intersecting with the object. This target contact point was then perturbed by a random angle of up to $\pm 10^\circ$ degrees (Fig. 9). The robot finger then moved along this straight line, for a distance $l$, at constant speed. During the push, video and finger position were captured at 30 Hz, including a 1 second buffer at either end of the finger trajectory. The object pose was tracked at 30 Hz, using a structural and texture edge based particle filter that is able to track in clutter and occlusion (Mörwald et al. 2009). Learning used the recovered pose of the object in every other frame (i.e. at 15 Hz). For test trials the trajectory of the finger was known a priori and the object pose observed only at the initial frame. Using these a trajectory was predicted for the object over 150 steps at 15 Hz (10 s). The predicted trajectory was created purely using the forward model, and did not use any observations during the push to update its prediction.[9] For real experiments, a 5-axis robot arm with a single finger was used. Simulation experiments used the NVIDIA PhysX engine (NVIDIA PhysX 2009) to provide ground-truth. The PhysX engine was evaluated as a predictor itself in real object trials. All methods required the predictor to know the object shape model, represented as a mesh, and the starting pose. The correct model (context) was determined by fitting models from a library to the visual data. The visual context detector automatically picked the model and pose with the best visual fit according to our particle filter (Mörwald et al. 2009), converging to $\pm 2$ mm error in the first few frames.

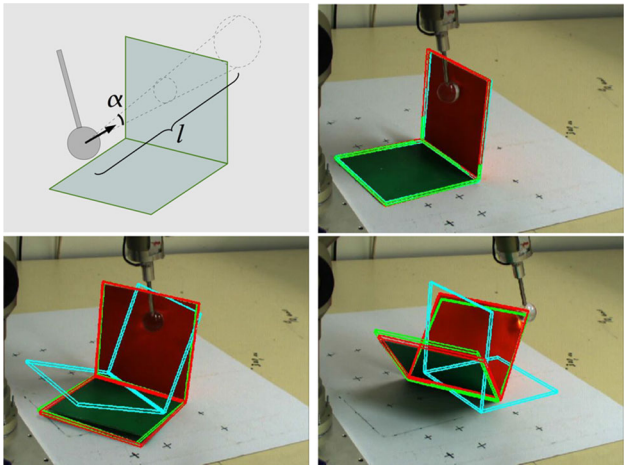



**Fig. 9** In all experiments the robot finger pushes in a straight-line of length $l$=25± 5 cm within a cone of angle $\alpha$=20 deg toward an object (*top left*). The start point is randomised so that each region on the vertical face is equally likely to be pushed. The *red* wire-frame shows the output of the visual tracker, the *green* wire-frame the object pose predicted by the KDEF learner, and the *blue* wire-frame the prediction of the PhysX simulator (Color figure online)

Local frames for environment contacts in the -GAE variants were fixed by hand to the edges of objects. In test cases with new objects the frames were again fixed by hand. An item for future work is to perform this process automatically. All methods required parameter tuning. Model selection was performed in experiment P1 to establish reasonable parameter values, which were then used in experiments P2 and P3. It was not possible to perform fully systematic optimisations for LWPR, KDE and KDEF, due to the size of the parameter spaces. Rather, subsets of the parameter space were selected by inspection, and then explored using grid search. Models were evaluated on a separate hold-out set, of the same size as the test set. Model selection by full grid search was performed for the following parameters of the PhysX simulator: static friction, dynamic friction, and the coefficient of restitution.

[9] This would undoubtedly improve performance, but would reduce the problem to one-step prediction, which is not the main subject of this paper.

**Table 3** Parameter settings for optimisation of PhysX

| Parameter | Values | | | | |
|---|---|---|---|---|---|
| restitution | 0.125 | 0.1875 | 0.25 | 0.375 | 0.5 |
| static friction | 0.25 | 0.3125 | 0.5 | 0.75 | 1.0 |
| dynamic friction | 0.25 | 0.3125 | 0.5 | 0.75 | 1.0 |

The full set of parameters is given in Table 3, leading to 125 different parameter combinations being tried for PhysX for each training object. PhysX had access to full mesh and contact information. Parameter search for the KDE methods was performed for the bandwidth of the kernels, with three values tried. For the KDE and LWPR methods three different parameterisations [Gauss-Euler (e), Gauss-Quaternion (q), Von-Mises-Fisher-Quat (v)] were studied in experiment P1, and the best was used in experiments P2 and P3. For LWPR we used the Euler parameterisation throughout experiments P1, P2, and P3. For experiment P1 we performed 10-fold cross-validation. The sizes of the training and test sets are stated in the method for each experiment. For transfer learning experiments P2 and P3, disjoint training and test sets were used. For clarity, a complete set of acronyms for the algorithm-information combinations is given in Table 2.

### 9.3 Performance measure

In all experiments with real objects, predicted trajectories were evaluated against the visually tracked object pose. The tracker does not provide perfect ground-truth, yielding errors of ±2mm. Prediction performance was evaluated as follows.

At any particular time step, $t$, a large number, $N$, of randomly chosen points $p_n^{1,t}$, where $n = 1 \ldots N$, are rigidly attached to an object at the ground-truth pose, and the corresponding points $p_n^{2,t}$ to an object at the predicted pose. At time step $t$, an average error $E_t$ can now be defined as the mean of displacements between points on the object at the predicted pose and points on the object at the ground-truth pose:

$$E_t = \frac{1}{N} \sum_{n=1 \ldots N} |p_n^{2,t} - p_n^{1,t}| \tag{21}$$

Note that, for each push action, we predict approximately 150 consecutive steps into the future, with no recursive filtering or corrector steps, hence it is expected that errors will grow with range from the initial object pose. We therefore find it more meaningful to normalise all errors with respect to an "average range", $R_t$, of the object from its starting position, defined as:

$$R_t = \frac{1}{N} \sum_{n=1 \ldots N} |p_n^{1,t} - p_n^{1,0}| \tag{22}$$

For a test data set, consisting of $K$ robotic pushes, each of which breaks down into many consecutive predictions over $T$ time steps, we can now define average error and normalised average error. Note that the normalised error measure necessarily has no units.

$$E_{av} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{T} \sum_{t=1}^{T} E_t, \quad E_{av}^{norm} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{T} \sum_{t=1}^{T} \frac{E_t}{R_t} \tag{23}$$

## 10 Results

### 10.1 Experiment P1: action interpolation

In Experiment P1 the robot applied a set of random pushes to a polyflap, a box and a cylinder respectively. All the algorithm variants in Table 2 were trained and tested. Model selection was performed for all algorithm-information combinations, including PhysX. Ten fold cross-validation was performed for all algorithms. The density estimation techniques were studied with all three parameterisations of rotation. Training (and testing) sets were 200 (25) pushes (cylinder), 400 (50) pushes (box) and 700 (90) pushes (polyflap). Figure 10 (left column) shows convergence of the best learning algorithms. Figure 10 (right column) shows how performance varies with input information for the best parameterisations of all the algorithms. Table 4 shows the results of model selection on the different parameterisations for KDE. Image sequences of predicted vs actual trajectories are shown in (Fig. 11). For the simulation experiments on non-planar surfaces, 900 (100) pushes of a cylinder in two different starting positions (upright and on its side) were made. For this simulated experiment we only used factored KDE with the Gaussian-quaternion parameterisation, on the basis that this was the best performing setup for the real experiments.

#### *10.1.1 Experiment P1 discussion*

Table 4 and Fig. 10 show that the learned models almost always outperformed physics simulation on the test set, with approximately one third the prediction error. Thus we find strong support for hypothesis H1. Regarding the parameterisation, Gaussian kernels with quaternions were best in 14 of 15 cases (Table 4 bold entries). Thus this parameterisation was used in experiments P2 and P3.

In Fig. 11 it can be seen that predictions were accurate and physically plausible for a variety of learning methods, even over 150 steps. Note that the physics simulator predicts incorrect turning of the cylinder when pushed (Fig. 11 column 8). Figure 10 shows that additional information A or E gives no advantage for any algorithm in this experiment, indeed

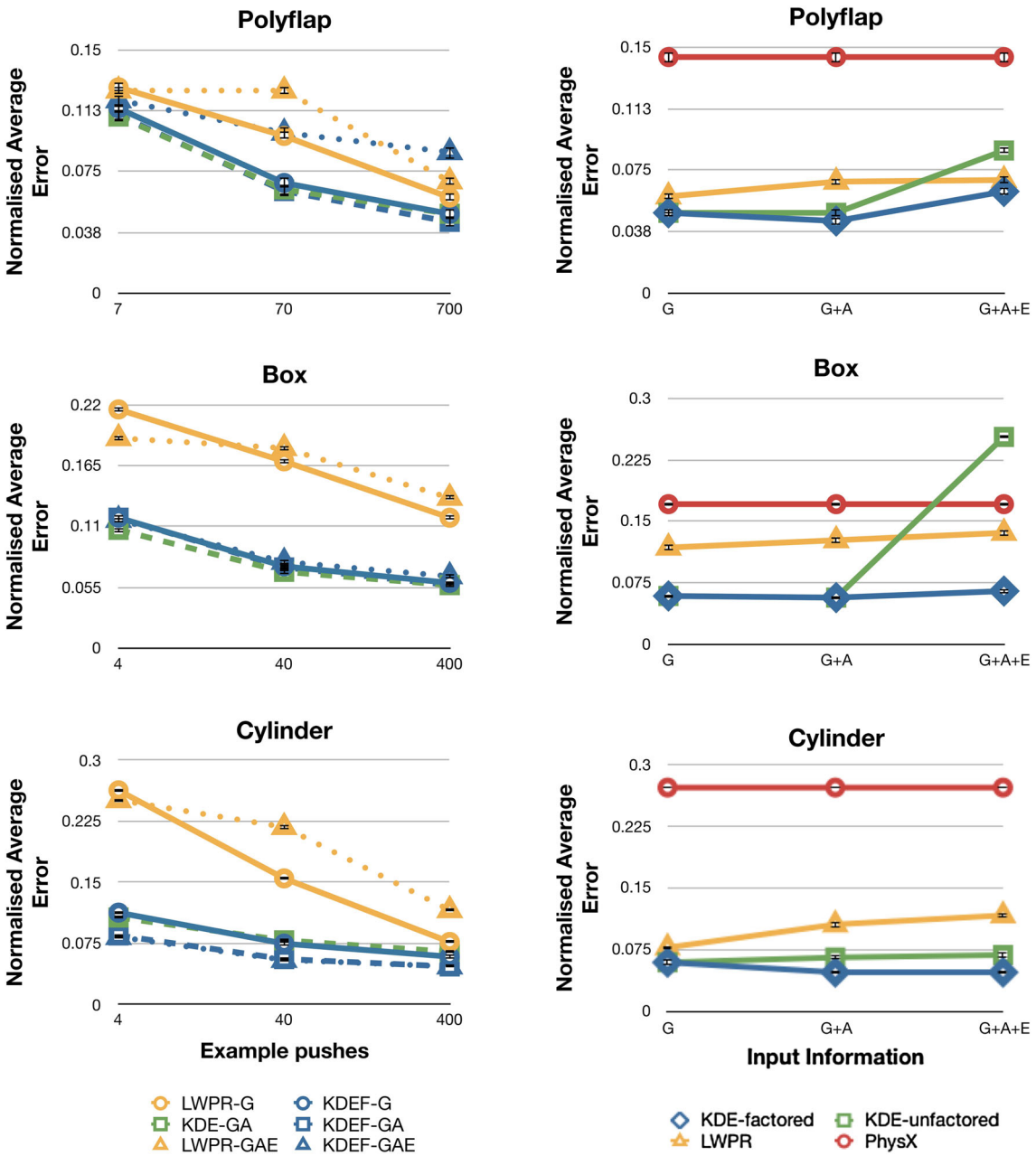



**Fig. 10** Experiment P1: Convergence of a selection of learning algorithm-information combinations (*left column*). Change in normalised average prediction errors with varying input information (*right column*). *G* global information, *A* agent-object information, *E* object-environment information. Standard *error bars* are shown in *black*, in most cases these are very small and fall within the symbol for each data point. We show *error bars* for all data points except for the learning convergence graph shown here (Color figure online)

LWPR gets worse with more dimensions. This is in line with expectation, since no learning transfer is being attempted. Finally, Figs. 12 and 13 show that the approach is also able to learn predictive models for a cylinder in a variety of starting positions on an uneven surface. This includes predictions of flipping the object up, and the object rolling away once contact is lost.

### 10.2 Experiment P2: action transfer

Experiment P2 tests hypothesis H2: whether predictions can be transfered to novel actions. The training set was 900 pushes applied to an L shaped flap in one direction (Fig. 6 top left). The test set was 100 pushes applied from the other side (Fig. 6 top right). The same method was followed in simula-

**Table 4** Experiment P1: forward push on a polyflap/box/cylinder, trained on real data

| Predictor | Polyflap | Box | Cylinder |
|---|---|---|---|
| KDEF-Ge | 0.055±0.002 | 0.061±0.003 | 0.063±0.003 |
| KDEF-Gq | **0.049**±0.002 | **0.059**±0.002 | **0.059**±0.003 |
| KDEF-Gv | 0.057±0.002 | 0.066±0.003 | 0.071±0.003 |
| LWPR-Ge | 0.059±0.002 | 0.118±0.003 | 0.077±0.003 |
| KDEF-GAe | 0.054±0.002 | 0.060±0.003 | 0.052±0.002 |
| KDEF-GAq | **0.044**±0.002 | **0.057**±0.002 | **0.047**±0.002 |
| KDEF-GAv | 0.064±0.002 | 0.097±0.002 | 0.109±0.003 |
| LWPR-GAe | 0.068±0.002 | 0.127±0.003 | 0.105±0.002 |
| KDEF-GAEe | 0.083±0.003 | 0.065±0.003 | 0.050±0.002 |
| KDEF-GAEq | **0.062**±0.002 | **0.065**±0.003 | **0.047**±0.002 |
| KDEF-GAEv | 0.081±0.002 | 0.086±0.002 | 0.065±0.002 |
| LWPR-GAEe | 0.069±0.002 | 0.136±0.003 | 0.116±0.003 |
| KDE-GAe | 0.053±0.002 | 0.057±0.002 | 0.068±0.003 |
| KDE-GAq | **0.049**±0.002 | **0.057**±0.002 | **0.065**±0.003 |
| KDE-GAv | 0.062±0.002 | 0.058±0.002 | 0.092±0.004 |
| KDE-GAEe | 0.090±0.002 | 0.161±0.003 | 0.071±0.003 |
| KDE-GAEq | **0.087**±0.002 | 0.253±0.002 | **0.068**±0.003 |
| KDE-GAEv | **0.087**±0.002 | **0.127**±0.003 | 0.091±0.004 |
| PhysX | 0.144±0.003 | 0.171±0.003 | 0.271±0.001 |

Shown is the dimensionless measure normalised average error $E_{av}^{norm}$± standard error. The parameterisations are denoted by e (Euler), q (Gauss Quaternion) and v (Gauss+Von-Mises Fisher) respectively
Bold values denote the parameterization of rotation (e, q or v) with the minimum error for each algorithm-information combination

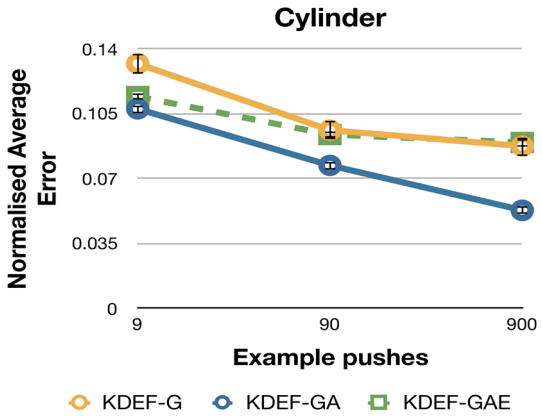



**Fig. 12** Experiment P1: Convergence of learning for a cylinder on an uneven surface

tion and with the real object. All the algorithm-information variants in Table 2 were tested. We measured the transfer prediction error, i.e. the prediction error for the novel test actions. Figure 14 shows the normalised average error $E_{av}^{norm}$ for the simulation experiment (left panel) and with real objects (right panel). Figure 15 shows example predicted trajectories on synthetic and real test cases.

### *10.2.1 Experiment P2 discussion*

Figure 14 (left panel) shows that, in simulation, additional contact information (A or AE) didn't improve performance

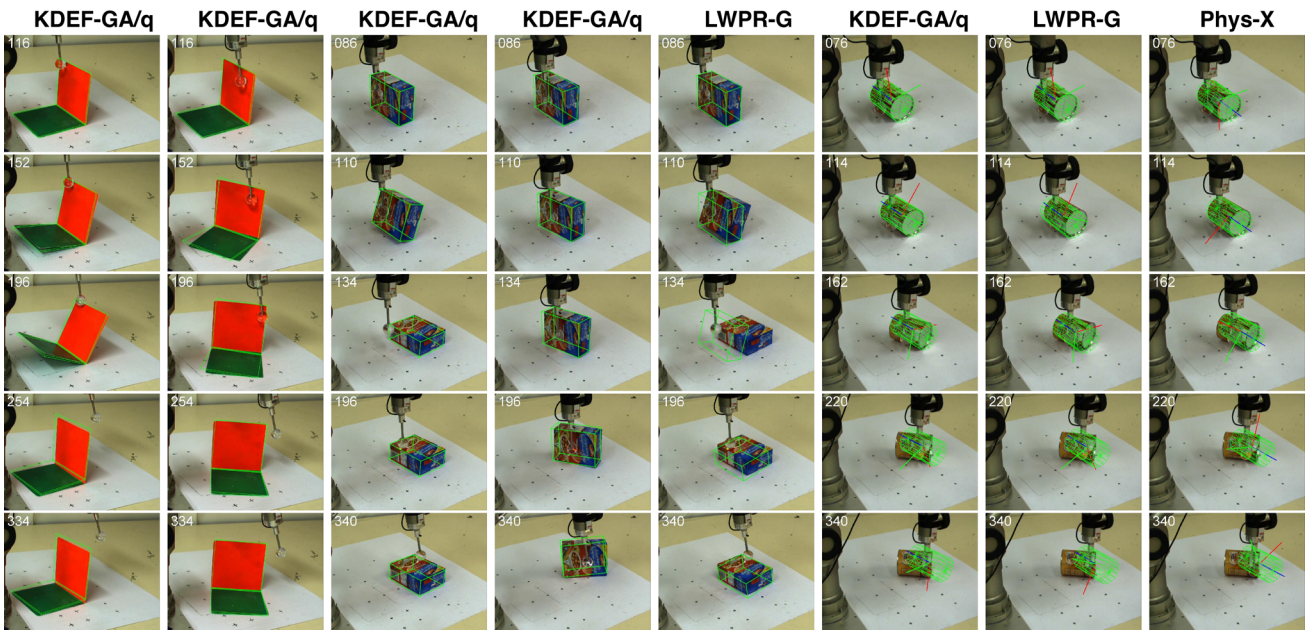



**Fig. 11** Experiment P1: polyflap, box and cylinder. *Green* outline shows predictions. *Columns 1–2* KDEF-GA/quat on two trials exhibiting different motions of the polyflap. *Column 3* KDEF-GA/quat. *Column 4* KDEF-GA/quat on another trial in which the *box* slides. *Column 5* LWPR-G for one trial in which the box topples over. *Columns 6–8*show the same push of the cylinder. *Column 6* KDEF-GA/quat. *Column 7* LWPR-G. *Column 8* PhysX. Note tha for *columns 6–7* the orientation of the cylinder is show by the rotating frame. Only the learned models predict the rotation correctly. Frame numbers are in the *top left* of each image

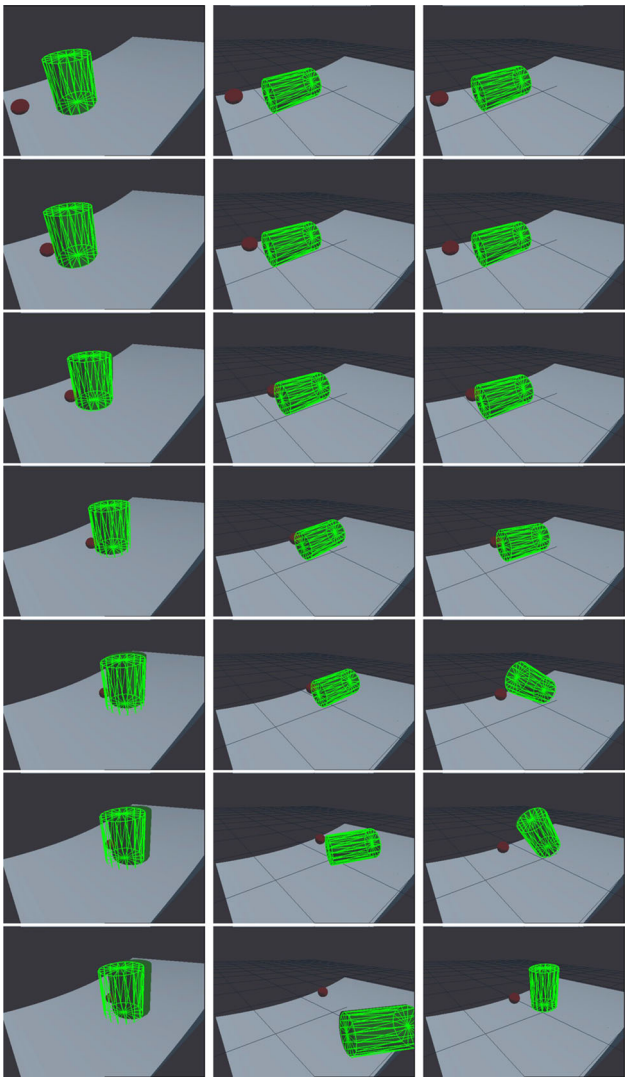


**Fig. 13** Experiment P1: predictions for a simulated cylinder on a non-planar surface. Predictions were made over 150 steps by the KDEF-GA/quat method. Note that the learned model can correctly predict for two starting orientations, and for dynamic flipping and rolling motions once the finger loses contact

of KDE and LWPR. In contrast, factorisation could take advantage of the additional information: the performance of KDEF improved significantly. The predictions of KDEF (Fig. 15) precisely match the hypothesized effects of adding contact information depicted in Fig. 6 (bottom row). With only global information the finger was predicted by KDEF to pass through the object (Fig. 15 column 1). By adding the agent-object information the prediction of KDEF was that the object would move with the finger, but that it would also penetrate the table (Fig. 15 column 2). By also adding object-environment information KDEF predicts that the object will slide along the table in contact with the finger (Fig. 15 column 3). On real objects (Fig. 14 right panel) the learned predictors slightly outperform the physics engine, and prediction accuracy declines. Additional contact information with factoring still enables KDEF to make physically plausible predictions. Figure 15 (columns 4 and 7) shows that KDEF-G and LWPR-G predict that the finger passes through the object, and that the object doesn't move. In Fig. 15 (columns 5 and 6) KDEF-GA correctly predicts the sliding motion of the object. Only factoring enables this, the unfactored methods don't produce plausible predictions. This supports hypothesis H2: factoring enables action transfer. Transfer performance is best if the training observations are accurate, diminishing with training noise.

### 10.3 Experiment P3: shape transfer

Experiment P3 tests hypothesis H3: can predictors that have been learned from one set of objects transfer their predictions to an object of novel shape? The experiment was run in simulation and with real objects. Shape transfer was tested from (i) a polyflap to a box (P3.A) and (ii) a box and a cylinder to a double cylinder (P3.B). Frame positions on training and test objects were determined using the heuristic procedure described earlier, resulting in placements as shown in Fig. 16. There were 900 training pushes on the polyflap and 200 test pushes on the box for (i), and 200 training pushes (100 box, 100 cylinder) and 100 test pushes (double cylinder) for (ii). This experiment ran on real objects for i) and ii) and in simulation for (i), giving three train-test conditions in total. All algorithm-information combinations in Table 2 were tried. When learning from two objects the same number of factors (experts) were used for each object, and they were matched across the two objects by hand. Thus each expert received a mix of data from each object, learning to encode rolling, sliding, or tipping motions. The normalised average error for all three conditions, $E_{av}^{norm}$, is shown in Fig. 17. Example frames are shown in Fig. 18.

#### *10.3.1 Experiment P3 discussion*

Shape transfer only occurs with contact information and factoring, such as for the KDEF-GA method in experiment P3.A (Fig. 18 column 6). Learners with global information predict that the finger passes through the box (Fig. 18 columns 5 and 7). In experiment P3.A only factoring plus all the contact information (KDEF-GAE) produced physically plausible predictions. In Fig. 18 (column 3) KDEF-GAE predicts that the double cylinder will slide along the table, whereas KDEF-G and KDEF-GA predict it will penetrate the table (Fig. 18 columns 1 and 2). KDEF-GAE also makes physically plausible predictions for a novel real object (Fig. 18 column 4). None of the other learners could achieve this. Only by using factoring plus all contact information was shape transfer learning achieved. In fact KDEF-GAE also matched the accuracy of the physics simulator. Thus this experiment

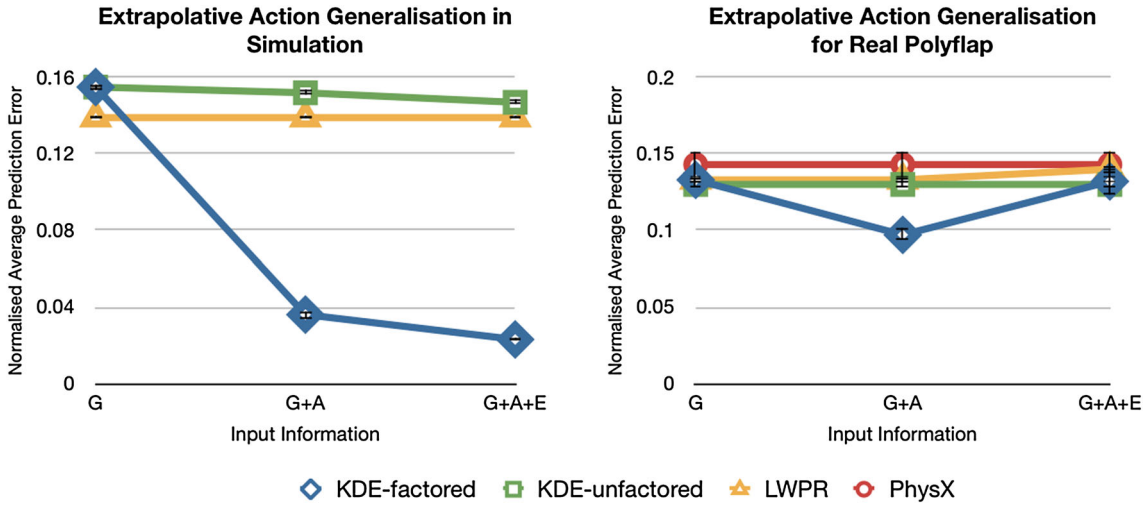



**Fig. 14** Experiment P2: Action transfer. Trained on forward push on polyflap, tested on backward push, for simulated (*top*) and real data (*bottom*). Comparative performance of predictors vs. information utilised (global/agent/environment), as measured by normalised average error $E_{av}^{norm}$

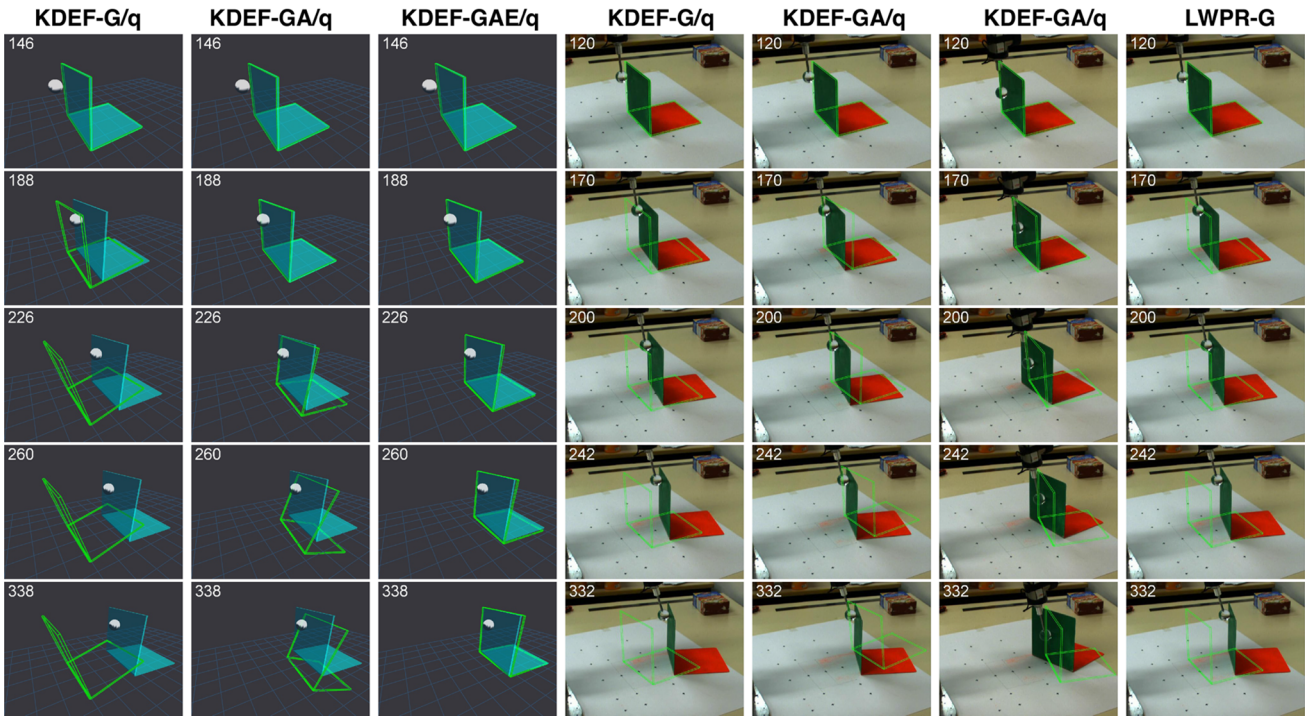



**Fig. 15** Experiment P2: Green outline shows predictions. *Column 1* KDEF-G/quat. *Column 2* KDEF-GA/quat. *Column 3* KDEF-GAE/quat. *Column 4* KDEF-G/quat. *Column 5* KDEF-GA/quat. *Column 6* KDEF-GA/quat. *Column 7* LWPR-G. Note that the KDEF-G/quat and LWPR-G methods predict that the robot finger passes through the polyflap. Frame numbers are in the *top left* of each image

supports hypothesis H3: factoring plus contact information enables shape transfer learning.

### 10.4 General discussion

Two questions arise from the experiments. First, why does PhysX fail to do better in P1, even though separately tuned to each object? The answer is that real objects don't adhere to its idealised friction model. This is a problem for all such simulators, so modular learning will always be better. Second, why does learning transfer performance decline on real data in P2 and P3? The cause is the tracking noise in the training data, which leads to perceived object-finger penetrations. This gives some probability, in the learned model, that the finger can pass through the object.

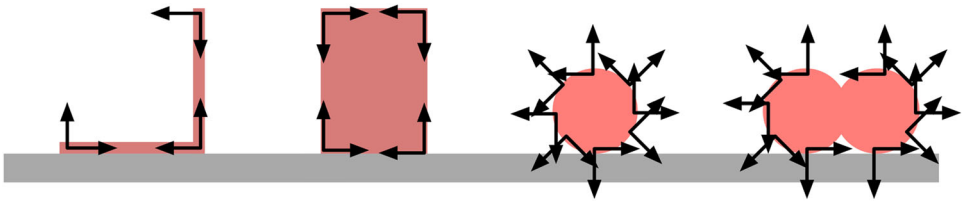


**Fig. 16** Experiment P3: Object-environment frame placements on the training and test objects. The training objects had the frames placed by hand. Frames on the test objects were selected to have similar local curvature. The cylinders had many more frames attached than possible to show here, placed at even intervals on the circumference

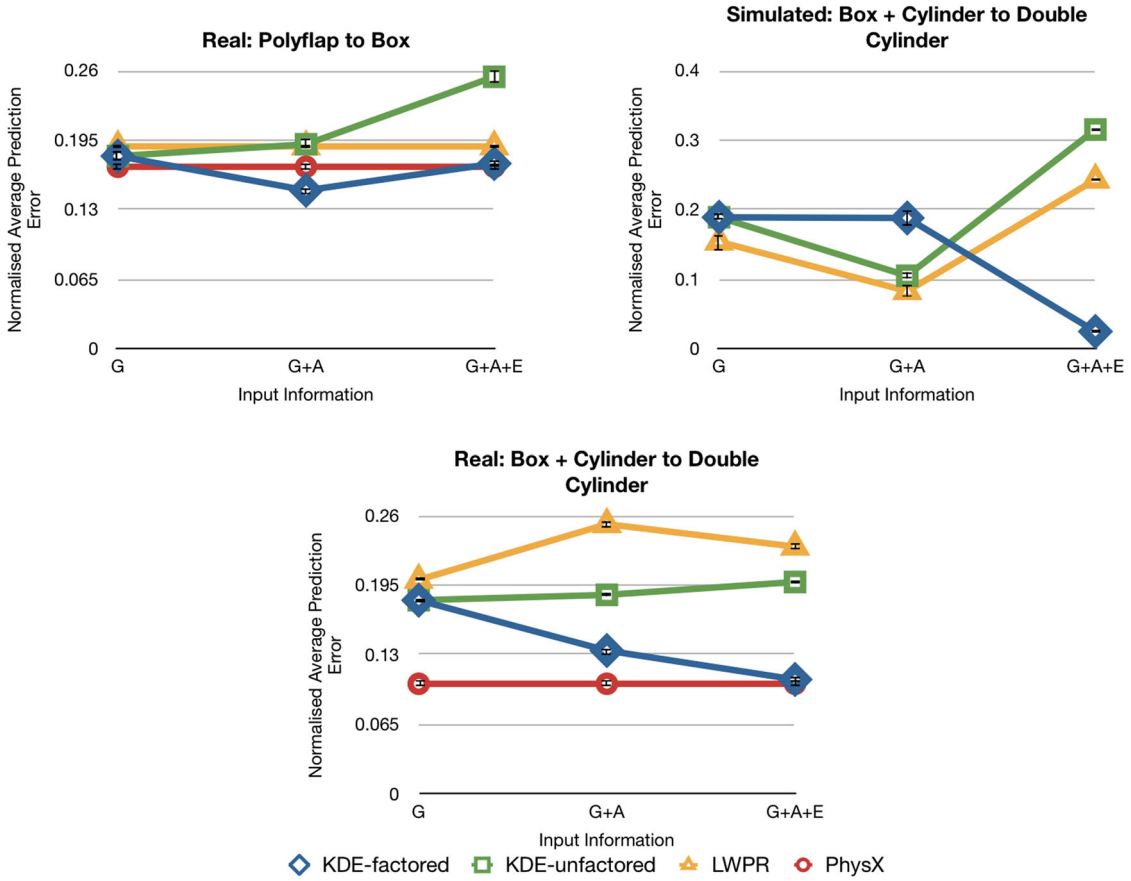



**Fig. 17** Experiment P3: Comparative performance of predictors vs. information utilised, as measured by the normalised average error $E_{av}^{\text{norm}}$

## 11 Related work

Related work is split into four areas: neuroscience, analytic models, qualitative physics, and machine learning. Prediction of the effects of motor actions has long been studied in neuroscience (Miall and Wolpert 1996; Flanagan et al. 2003). MOSAIC was an early computational model of prediction and control in the cerebellum using a modular scheme (Haruno et al. 2001), where predictions can be made by convex combinations of learned predictors. Other bio-inspired modular prediction schemes were independently derived by roboticists (Demiris and Khadhouri 2006). These models all differ from ours, in that our work is the first attempt at learning to model the motions of objects with kinematic constraints. Developmentally, there is evidence that infants can learn object specific motions (Bahrick and Pickens 1995). It is also clear that while some object knowledge may be innate (Spelke et al. 1994), object specific predictions must be learned, and are critical to our manipulation skills (Flanagan et al. 2006). So, in general terms, modular learning of predictions of object behaviour is cognitively plausible.

There is substantial work in robotics on analytic models of pushing (Mason 1982; Lynch 1992; Peshkin and Sanderson 1988; Cappelleri et al. 2006), including both kinematic

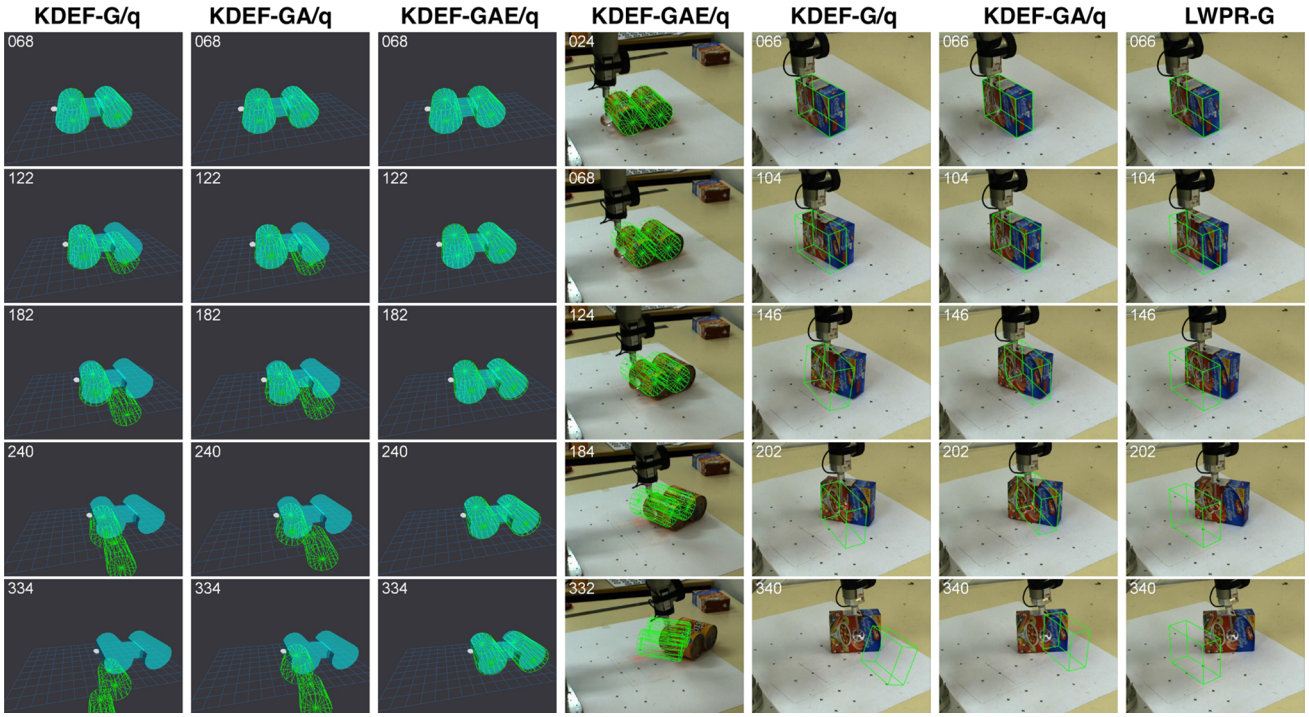



**Fig. 18** Experiment P3: Shape Transfer. *Green* outline shows predictions. *Column 1* KDEF-G/quat. *Column 2* KDEF-GA/quat. *Column 3* KDEF-GAE/quat. *Column 4* KDEF-GAE/quat. *Column 5* KDEF-G/quat. *Column 6* KDEF-GA/quat. *Column 7* LWPR-G, all for one trial. Note that the KDEF-G/quat and LWPR-G methods predict that the robot finger moves into the *box*. Frame numbers are in the *top left* of each image

and dynamic models of manipulation effects (Mason 2001). Analytic dynamics models are also used to model tasks involving switching dynamics between flight and impact, such as juggling (Brogliato and Río 2000). Such analytic models are good metric predictors if their key parameters (e.g. friction) are precisely known, although qualitative predictions are robust to parameter uncertainty. They can also inform push planning under pose uncertainty (Brost 1985). These approaches are appealing in that proofs concerning the qualitative object motion can be obtained, particularly under quasi-static conditions (Mason and Salisbury 1985; Peshkin and Sanderson 1988). This led to methods for push planning that have some guarantees, such as completeness and optimality (Lynch and Mason 1996). A related approach to full physics simulation is the use of generic physics principles, such as the principle of minimum work, to make precise predictions (Kopicki et al. 2010). There is a separate body of work on qualitative models of action effects on objects, rooted in naive physics (Hayes 1995) and qualitative physics (Kuipers 1986). In a similar spirit, there is work on using physics engines to learn qualitative action effects (Mugan and Kuipers 2012), and on high level planning of manipulation (Stilman and Kuffner 2008; Roy et al. 2004) using qualitative action models. Some early ideas on push planning have reappeared in recent robots, which plan pushes to enable grasps in clutter (Dogar and Srinivasa 2010).

Learning for forward modelling has been long understood (Jordan and Jacobs 1990; Jordan and Rumelhart 1992). In robotics it has been used to model contactless motion, e.g. predicting the motion of an object or robot manipulator in free space (Ting et al. 2006; Boots et al. 2014; Dearden and Demiris 2005). It has also been used to learn the dynamics of an object with a single, constant contact (such as pole balancing) (Schaal 1997; Atkeson and Schaal 1997). Finally, there has been work on affordance learning, and work on identifying which variables are relevant to predicting object motion (Montesano et al. 2008; Moldovan et al. 2012; Hermans et al. 2011; Fitzpatrick et al. 2003; Ridge et al. 2010; Kroemer and Peters 2014). The restriction of these papers is that they make qualitative predictions of object motion, such as a classification of the type of motion outcome. There has also been work on predicting stable push locations (Hermans et al. 2013). Others have worked on learning metric motion models from experience. Stoytchev (Stoytchev 2008) enabled a robot to learn action effects of sticks and hook-like tools by pushing objects. This work simplifies the domain by using circular pucks as objects, and four planar motions as actions. Action outcomes were learned for various tools in a modular fashion, but without transfer learning. In both (Meriçli et al. 2014) and Scholz and Stilman (2010) the metric planar motion of pushed objects on the plane is learned, and the learning is modularized by object, as here. In each case a small number

of discrete pushing actions is tried, and motion models are planar, rather than full rigid body transformations.

Modelling of contacts is also useful for other tasks. For example, both learning and analytic approaches to prediction have been used for visual tracking (Mörwald et al. 2011; Duff et al. 2011; Pham et al. 2015). Finally, modelling contacts with products of experts has also been used for object grasping (Kopicki et al. 2015, 2014) and placement (Kopicki and Wyatt 2015).

Our work sits at the intersection of some these approaches. We embrace machine learning and modularity to achieve scalability, but we also explicitly model each contact. Our machine learning approach is used to make metrically precise predictions, under contact, including changing contact with the environment. In this way we try to re-achieve in a machine learning framework what only the analytic approach has attempted to date: metric predictions of motion that are transferable to novel actions and objects.

## 12 Conclusions

This paper found that modular predictors of object motion can be learned; that learning transfer is possible; that contact information assists transfer; that factorisation helps to exploit this information; and that learning can match, or even exceed, physics engine performance. The paper presented the first results on real objects for object transfer. What do these results tell us about the way to proceed? We note the following issues.

### 12.1 Prior knowledge

While the prior knowledge embodied by classical mechanics provides generality, the necessary approximations made in implementations can hinder accurate prediction. Rigid body simulators also require learning of the intrinsic parameters of the object, but sometimes have too many constraints to wrap themselves finely around real data. On the other hand, it is clear that some structural knowledge is required: contact information is structural knowledge benefiting transfer. Pure tabula rasa learning is unlikely to be the answer.

### 12.2 Local shape

The learners employed here used much less information than the full object shape. Further shape information might improve prediction further. Specifically, the local surface shapes of both surfaces at a contact influence object motion. Experts specialised to local shape contexts may improve prediction performance. In the scheme presented here, this would result in nesting another modular structure inside the product of experts. It would also provide a means to solve the problem of how to automatically attach experts to objects.

### 12.3 Modularity

There is evidence that the brain employs modularity in prediction and control. We have argued that this is a promising way to proceed for robotics. Rather than learning a general purpose predictor, why not learn many specific predictors? Memory in current computing technology is cheap, and so learning many hundreds, or even thousands, of object specific prediction modules is feasible. Modularity is part of the way to proceed.

### 12.4 Multiple changing contacts

In manipulation, the hand makes multiple changing contacts with the object. Prediction for manipulation must account for these non-smooth changes. Hybrid models may be a way to proceed. These have been explored in modelling changing contact dynamics in walking, but have yet to be applied to manipulation.

### 12.5 Training noise

Transfer performance degrades under training noise. Recently, we have partially addressed this by removing noise at prediction time, using kinematic optimisation (Belter et al. 2014). This combines the benefits of collision checking and machine learning, improving prediction performance significantly. The collision checker thus implements the commonsense physics rule that rigid bodies can't interpenetrate. Whether these initial results are extensible is a topic for future work.

### 12.6 Dynamics

We have restricted this study to quasi-static cases, but the formulation of the basic regression problem with dynamics was given. Learning dynamics is the next step.

**Acknowledgments** We gratefully acknowledge support of Grant EU-FP7-IST-600918-PaCMan.

**Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

## References


Abramowitz, M., & Stegun, I. A. (1965). *Handbook of Mathematical Functions*. New York: Dover.

Atkeson, C.G., & Schaal, S. (1997). Learning tasks from a single demonstration. In *1997 IEEE international conference on robotics and automation* (Vol. 2, pp. 1706–1712).

Bahrick, L. E., & Pickens, J. N. (1995). Infant memory for object motion across a period of three months: Implications for a four-phase attention function. *Journal of Experimental Child Psychology*, *59*(3), 343–371.

Belter, D., Kopicki, M., Zurek, S., & Wyatt, J. (2014). Kinematically optimised predictions of object motion. In *Proceedings of the IEEE/RSJ international conference on intelligent robotics and systems (IROS 2014)* (pp. 4422–4427).

Boots, B., Byravan, A., & Fox, D. (2014). Learning predictive models of a depth camera and manipulator from raw execution traces. In *Proceedings of the IEEE international conference on robotics and automation (ICRA 14)* (pp. 4021–4028).

Brogliato, B., & Río, A. Z. (2000). On the control of complementary-slackness juggling mechanical systems. *IEEE Transactions on Automatic Control*, *45*(2), 235–246.

Brost, R. C. (1985). Planning robot grasping motions in the presence of uncertainty. Tech. Rep. CMU-RI-1R-85-12.

Cappelleri, D.J., Fink, J., Mukundakrishnan, B., Kumar, V., & Trinkle, J. C. (2006). Designing open-loop plans for planar micro-manipulation. In *Proceedings of the IEEE international conference on robotics and automation (ICRA 2006)* (pp. 637–642).

Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *17*(8), 790–799.

Cosgun, A., Hermans, T., Emeli, V., & Stilman, M. (2011). Push planning for object placement on cluttered table surfaces. In *Proceedings of the IEEE/RSJ international conference on intelligent robotics and systems (IROS 2011)* (pp. 4627–4632).

Craik, K. J. W. (1943). *The nature of explanation*. CUP Archive.

Dearden, A., & Demiris, Y. (2005). Learning forward models for robots. In: *Proceedings of the international joint conference on artificial intelligence (IJCAI 2005)* (Vol. 5, p. 1440).

Demiris, Y., & Johnson, M. (2003). Distributed, predictive perception of actions: A biologically inspired robotics architecture for imitation and learning. *Connection Science*, *15*(4), 231–243.

Demiris, Y., & Khadhouri, B. (2006). Hierarchical attentive multiple models for execution and recognition of actions. *Robotics and Autonomous Systems*, *54*(5), 361–369.

Detry, R. (2010). Learning of multi-dimensional, multi-modal features for robotic grasping. Ph.D. thesis, University of Liege.

Dogar, M., & Srinivasa, S. (2010). Push-grasping with dexterous hands: Mechanics and a method. In *Proceedings of the IEEE/RSJ international conference on intelligent robotics and systems (IROS 2010)* (pp. 2123–2130).

Duff, D.J., Mörwald, T., Stolkin, R., & Wyatt, J. (2011). Physical simulation for monocular 3d model based tracking. In *2011 IEEE international conference on robotics and automation (ICRA)* (pp. 5218–5225).

Fitzpatrick, P., Metta, G., Natale, L., Rao, S., & Sandini, G. (2003). Learning about objects through action-initial steps towards artificial cognition. In *Proceedings of the IEEE international conference on robotics and automation (ICRA 2003)* (Vol. 3, pp. 3140–3145).

Flanagan, J. R., Bowman, M. C., & Johansson, R. S. (2006). Control strategies in object manipulation tasks. *Current Opinion in Neurobiology*, *16*, 650–659.

Flanagan, J. R., Vetter, P., Johansson, R. S., & Wolpert, D. M. (2003). Prediction precedes control in motor learning. *Current Biology*, *13*, 146–150.

Flickinger, D. M., Williams, J., & Trinkle, J. C. (2015). Performance of a method for formulating geometrically exact complementarity constraints in multibody dynamic simulation. *Journal of Computational and Nonlinear Dynamics*, *10*(1), 011,010–011,010-12.

Haruno, M., Wolpert, D. M., & Kawato, M. (2001). Mosaic model for sensorimotor learning and control. *Neural Computation*, *13*, 2201–2220.

Hayes, P. J. (1995). The second naive physics manifesto. In G. F. Luger (Ed.), *Computation & intelligence* (pp. 567–585). Menlo Park, CA: AAAI.

Hermans, T., Li, F., Rehg, J. M., & Bobick, A. F. (2013). Learning contact locations for pushing and orienting unknown objects. In *Proceedings of the IEEE-RAS international conference on humanoid robotics* (pp. 435–442).

Hermans, T., Rehg, J., & Bobick, A. (2011). Affordance prediction via learned object attributes. In *ICRA 2011: Workshop on semantic perception, mapping, and exploration*.

Johansson, R. J., & Cole, K. J. (1992). Sensory-motor coordination during grasping and manipulative actions. *Current Opinion in Neurobiology*, *2*, 815–823.

Jordan, M. I., & Jacobs, R. A. (1990). Learning to control an unstable system with forward modeling. In D. S. Touretzky (Ed.), *Advances in neural information processing systems* (pp. 324–331). Cambridge, MA: MIT Press.

Jordan, M. I., & Rumelhart, D. E. (1992). Forward models: Supervised learning with a distal teacher. *Cognitive Science*, *16*, 307–354.

Klanke, S., Vijayakumar, S., & Schaal, S. (2008). A library for locally weighted projection regression. *The Journal of Machine Learning Research*, *9*, 623–626.

Kopicki, M.: (2010). Prediction learning in robotic manipulation. Ph.D. thesis, University of Birmingham.

Kopicki, M., Detry, R., Adjigble, M., Stolkin, R., Leonardis, A., & Wyatt, J. L. (2015). One-shot learning and generation of dexterous grasps for novel objects. *The International Journal of Robotics Research*. doi:10.1177/0278364915594244.

Kopicki, M., Detry, R., Schmidt, F., Borst, C., Stolkin, R., & Wyatt, J. L. (2014). Learning dexterous grasps that generalise to novel objects by combining hand and contact models. In *2014 IEEE international conference on robotics and automation (ICRA)* (pp. 5358–5365).

Kopicki, M., Stolkin, R., Zurek, S., Mörwald, T., & Wyatt, J. (2010). Predicting workpiece motions under pushing manipulations using the principle of minimum energy. In *Proceedings of the RSS workshop on representations for object grasping and manipulation in single and dual arm tasks, Zaragoza, Spain*.

Kopicki, M., Wyatt, J., & Stolkin, R. (2009). Prediction learning in robotic pushing manipulation. In *Proceedings of the international conference on advanced robotics (ICAR 2009)* (pp. 1–6).

Kopicki, M., & Wyatt, J. L. (2015). One shot contact learning. In *Proceedings of the RSS Workshop on bridging the gap between data-driven and analytical physics-based grasping and manipulation, Rome, Italy*.

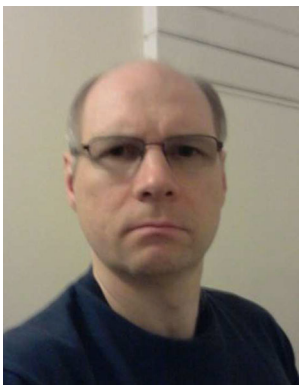
Kopicki, M., Zurek, S., Mörwald, T., & Wyatt, J. (2011). Learning to predict how rigid objects behave under simple manipulation. In *Proceedings of the IEEE international conference on robotics and automation (ICRA 2011)*.

Kroemer, O., & Peters, J. (2014). Predicting object interactions from contact distributions. In *Proceedings of the IEEE/RSJ international conference on intelligent robotics and systems (IROS 2014)*.

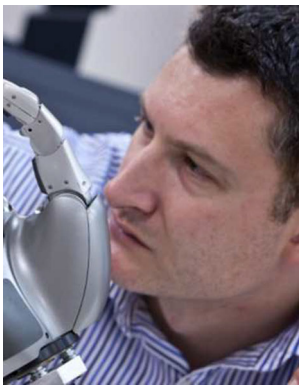
Kuipers, B. (1986). Qualitative simulation. *Artificial Intelligence Journal*, *29*(3), 289–338.

Lynch, K. (1992). The mechanics of fine manipulation by pushing. In *Proceedings of the IEEE international conference on robotics and automation (ICRA 1992)* (pp. 2269–2276).
Lynch, K. M., & Mason, M. T. (1996). Stable pushing: Mechanics, controllability, and planning. *International Journal of Robotics Research*, *15*, 533–556.
Mason, M. T. (1982). Manipulator grasping and pushing operations. PhD thesis, MIT.
Mason, M. T. (2001). *Mechanics of robotic manipulation*. Cambridge, MA: MIT press.
Mason, M. T., & Salisbury, J. K. (1985). *Robot hands and the mechanics of manipulation*. Cambridge, MA: The MIT Press.
Mehta, B., & Schaal, S. (2002). Forward models in visuomotor control. *Journal of Neurophysiology*, *88*, 942–953.
Meriçli, T., Veloso, M., & Akın, H. L. (2014). Push-manipulation of complex passive mobile objects using experimentally acquired motion models. *Autonomous Robots*, *38*(3), 317–329.
Miall, R., & Wolpert, D. (1996). Forward models for physiological motor control. *Neural Networks*, *9*(8), 1265–1279.
Moldovan, B., Moreno, P., van Otterlo, M., Santos-Victor, J., & Raedt, L. D. (2012). Learning relational affordance models for robots in multi-object manipulation tasks. In *Proceedings of the IEEE international conference on robotics and automation (ICRA 2012)* (pp. 4373–4378).
Montesano, L., Lopes, M., Bernardino, A., & Santos-Victor, J. (2008). Learning object affordances: From sensory-motor coordination to imitation. *IEEE Transations on Robotics*, *24*(1), 15–26.
Mörwald, T., Kopicki, M., Stolkin, R., Wyatt, J., Zurek, S., Zillich, M., & Vincze, M. (2011). Predicting the unobservable visual 3d tracking with a probabilistic motion model. In *2011 IEEE international conference on robotics and automation (ICRA)* (pp. 1849–1855).
Mörwald, T., Zillich, M., & Vincze, M. (2009). Edge tracking of textured objects with a recursive particle filter. In *Proceedings of the graphicon*.
Mugan, J., & Kuipers, B. (2012). Autonomous learning of high-level states and actions in continuous environments. *IEEE Transactions on Autonomous Mental Development*, *4*(1), 70–86.
Murray, R., Li, Z., & Sastry, S. (1994). *A mathematical introduction to robotic manipulation*. Boca Raton, FL: CRC Press.
NVIDIA PhysX: Physics simulation for developers (2009). http://developer.nvidia.com/physx.
Peshkin, M., & Sanderson, A. (1988). The motion of a pushed, sliding workpiece. *IEEE Journal on Robotics and Automation*, *4*, 569–598.
Pham, T. H., Kheddar, A., Qammaz, A., & Argyros, A. (2015). Capturing and reproducing hand-object interactions through vision-based force sensing. In: *IEEE international conference on computer vision workshops (OUI 2015 - ICCVW 2015)*.
Ridge, B., Skocaj, D., & Leonardis, A. (2010). Self-supervised cross-modal online learning of basic object affordances for developmental robotic systems. In: *Proceedings of the IEEE international conference on robotics and automation (ICRA 2010)* (pp. 5047–5054).
Roy, D., Hsiao, K. Y., & Mavridis, N. (2004). Mental imagery for a conversational robot. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *34*(3), 1374–1383.
Schaal, S. (1997). Learning from demonstration. *Advances in Neural Information Processing Systems*, *9*, 1040–1046.
Scholz, J., & Stilman, M. (2010). Combining motion planning and optimization for flexible robot manipulation. In: *2010 10th IEEE-RAs international conference on humanoid robots (humanoids)* (pp. 80–85).
Scott, D. W., & Sain, S. R. (2004). *Multi-dimensional density estimation* (pp. 229–263). Amsterdam: Elsevier.
Spelke, E. S., Katz, G., Purcell, S. E., Ehrlich, S. M., & Breinlinger, K. (1994). Early knowledge of object motion: Continuity and inertia. *Cognition*, *51*(2), 131–176.
Stilman, M., & Kuffner, J. J. (2008). Planning among movable obstacles with artificial constraints. *International Journal of Robotics Research*, *27*(12), 1296–1307.
Storn, R., & Price, K. (1997). Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, *11*(4), 341–359.
Stoytchev, A. (2008). Learning the affordances of tools using a behavior-grounded approach. *Lecture Notes in Artificial Intelligence*, *4760*, 140–158.
Ting, J. A., Mistry, M., Peters, J., Schaal, S., & Nakanishi, J. (2006). A Bayesian approach to nonlinear parameter identification for rigid body dynamics. In *Proceedings of robotics: Science and systems*.
Vijayakumar, S., D'souza, A., & Schaal, S. (2005). Incremental online learning in high dimensions. *Neural Computation*, *17*(12), 2602–2634.
Witney, A. G., Goodbody, S. J., & Wolpert, D. M. (2000). Learning and decay of prediction in object manipulation. *Journal of Neurophysiology*, *84*(1), 334–343.
Zito, C., Stolkin, R., Kopicki, M., & Wyatt, J. L. (2012). Two-level RRT planning for robotic push manipulation. In: *Proceedings of the IEEE/RSJ international conference on intelligent robotics and systems (IROS 2012)* (pp. 678–685).

**Marek Kopicki** is a post-doctoral researcher at the University of Birmingham. He has a MSc in physics at Adam Mickiewicz University (Poznan), a MSc in Advance Computer Science and Ph.D. in robotics at University of Birmingham in 2010. He has more than twelve years' experience of developing stateof- the-art algorithms and software for robot control and vision which underpinned much of FP7 CogX, FP7 GeRT, and now is central to FP7 PaC-Man and H2020 RoMaNs. He invented an adaptive grasping algorithm, for which he holds an international patent pending. He has published a number refereed papers in robotics in conferences and top journals.



**Sebastian Zurek** is at the Department of Computer Science, University of Birmingham, UK. He has a BA degree in Theoretical Physics from the University of Cambridge, and was a researcher at the Cavendish Laboratory, where he obtained his PhD. He also has an MRes in Brain Imaging from the University of Birmingham.



**Rustam Stolkin** currently serves as the Senior Birmingham Fellow in robotics, at University of Birmingham, UK. Dr. Stolkin's research spans areas of science and engineering outside of robotics, as well as manipulation with robotic arms and hands, novel robotic vehicles, computer vision and other kinds of autonomous sensing, and a long-term interest in engineering and science education. He is involved in collaborations between academia and industry, with current applied projects including robotic decommissioning of nuclear waste, and robotics for bomb disposal and robotics for manufacturing. He is an active member of the IEEE RAS Technical Committee on Robotics and Automation in Nuclear Facilities.

**Thomas Moerwald** received a Ph.D. (Dr.Techn.) in Electrical Engineering at the Vision for Robotics group at the Vienna University of Technology (TUW) in 2013. He was part of the CogX project [FP7/2007-2013], the InModo project [FFG/836490] and is now within the FLOBOT project [H2020- ICT-2014-1/645 376]. His research interests are in the field of visual tracking, surface reconstruction using B- and T-splines, GPU computing, computer graphics and visualization.

**Jeremy L. Wyatt** is Professor of Robotics and Artificial Intelligence at the University of Birmingham. He has a BA in Theology (Bristol), an MSc in Artificial Intelligence (Sussex) and a Ph.D. in Machine Learning (Edinburgh, 1997). He has published more than 90 refereed papers, edited three books, received two best paper awards, supervised a BCS distinguished doctoral dissertation, and led a variety of research projects on artificial intelligence and robotics. He works on robot task planning, robot motion planning, robot manipulation, and robot learning in open, uncertain and unfamiliar environments.